
STM32F302xB/C/D/E and STM32F302x6/8
advanced ARM®-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32F302xB/C/D/E and STM32F302x6/8 microcontroller memory and peripherals. The STM32F302xB/C/D/E and STM32F302x6/8 devices are referred to as STM32F302xx throughout the document, unless otherwise specified.

The STM32F302xx is a family of microcontrollers with different memory sizes, packages and peripherals.

For ordering information, mechanical and electrical device characteristics please refer to the STM32F302xB/C, STM32F302xD/E and STM32F302x6/8 datasheets.

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

Related documents

- STM32F302xB/C, STM32F302xD/E and STM32F302x6/8 datasheets available from www.st.com
- STM32F3xx/F4xx ARM® Cortex®-M4 programming manual (PM0214) available from www.st.com.

Contents

1	Overview of the manual	40
2	Documentation conventions	42
2.1	List of abbreviations for registers	42
2.2	Glossary	42
2.3	Peripheral availability	43
3	System and memory overview	44
3.1	System architecture	44
3.1.1	S0: I-bus	46
3.1.2	S1: D-bus	46
3.1.3	S2: S-bus	46
3.1.4	S3, S4: DMA-bus	46
3.1.5	BusMatrix	47
3.2	Memory organization	48
3.2.1	Introduction	48
3.2.2	Memory map and register boundary addresses	48
3.3	Embedded SRAM	56
3.3.1	Parity check	56
3.4	Flash memory overview	56
3.5	Boot configuration	57
3.5.1	Embedded boot loader	57
4	Embedded Flash memory	58
4.1	Flash main features	58
4.2	Flash memory functional description	58
4.2.1	Flash memory organization	58
4.2.2	Read operations	59
4.2.3	Flash program and erase operations	62
4.3	Memory protection	68
4.3.1	Read protection	68
4.3.2	Write protection	70
4.3.3	Option byte block write protection	71

4.4	Flash interrupts	71
4.5	Flash register description	72
4.5.1	Flash access control register (FLASH_ACR)	72
4.5.2	Flash key register (FLASH_KEYR)	72
4.5.3	Flash option key register (FLASH_OPTKEYR)	73
4.5.4	Flash status register (FLASH_SR)	73
4.5.5	Flash control register (FLASH_CR)	74
4.5.6	Flash address register (FLASH_AR)	75
4.5.7	Option byte register (FLASH_OBR)	76
4.5.8	Write protection register (FLASH_WRP)	77
4.6	Flash register map	77
5	Option byte description	79
6	Cyclic redundancy check calculation unit (CRC)	82
6.1	Introduction	82
6.2	CRC main features	82
6.3	CRC functional description	83
6.4	CRC registers	84
6.4.1	Data register (CRC_DR)	84
6.4.2	Independent data register (CRC_IDR)	85
6.4.3	Control register (CRC_CR)	85
6.4.4	Initial CRC value (CRC_INIT)	86
6.4.5	CRC polynomial (CRC_POL)	86
6.4.6	CRC register map	87
7	Flexible memory controller (FMC)	88
7.1	FMC main features	88
7.2	Block diagram	89
7.3	AHB interface	90
7.3.1	Supported memories and transactions	90
7.4	External device address mapping	91
7.4.1	NOR/PSRAM address mapping	92
7.4.2	NAND Flash memory/PC Card address mapping	93
7.5	NOR Flash/PSRAM controller	94
7.5.1	External memory interface signals	95

7.5.2	Supported memories and transactions	97
7.5.3	General timing rules	98
7.5.4	NOR Flash/PSRAM controller asynchronous transactions	99
7.5.5	Synchronous transactions	116
7.5.6	NOR/PSRAM controller registers	123
7.6	NAND Flash/PC Card controller	129
7.6.1	External memory interface signals	130
7.6.2	NAND Flash / PC Card supported memories and transactions	132
7.6.3	Timing diagrams for NAND Flash memory and PC Card	132
7.6.4	NAND Flash operations	133
7.6.5	NAND Flash prewait functionality	134
7.6.6	Computation of the error correction code (ECC) in NAND Flash memory	135
7.6.7	PC Card/CompactFlash operations	136
7.6.8	NAND Flash/PC Card controller registers	138
7.7	FMC register map	145
8	Power control (PWR)	147
8.1	Power supplies	147
8.1.1	Independent A/D and D/A converter supply and reference voltage	148
8.1.2	Battery backup domain	148
8.1.3	Voltage regulator	149
8.2	Power supply supervisor	150
8.2.1	Power on reset (POR)/power down reset (PDR)	150
8.2.2	Programmable voltage detector (PVD)	151
8.3	Low-power modes	152
8.3.1	Slowing down system clocks	152
8.3.2	Peripheral clock gating	153
8.3.3	Sleep mode	153
8.3.4	Stop mode	154
8.3.5	Standby mode	156
8.3.6	Auto-wakeup from low-power mode	157
8.4	Power control registers	159
8.4.1	Power control register (PWR_CR)	159
8.4.2	Power control/status register (PWR_CSR)	160
8.4.3	PWR register map	162

9	Peripheral interconnect matrix	163
9.1	Introduction	163
9.2	Connection summary	163
9.3	Interconnection details	166
9.3.1	DMA interconnections	166
9.3.2	From ADC to ADC	166
9.3.3	From ADC to TIM	166
9.3.4	From TIM and EXTI to ADC	167
9.3.5	From OPAMP to ADC	167
9.3.6	From TS to ADC	167
9.3.7	From VBAT to ADC	167
9.3.8	From VREFINT to ADC	168
9.3.9	From COMP to TIM	168
9.3.10	From TIM to COMP	169
9.3.11	From DAC to COMP	169
9.3.12	From VREFINT to COMP	170
9.3.13	From DAC to OPAMP	170
9.3.14	From TIM to OPAMP	170
9.3.15	From TIM to TIM	170
9.3.16	From break input sources to TIM	171
9.3.17	From HSE, HSI, LSE, LSI, MCO, RTC to TIM	171
9.3.18	From TIM and EXTI to DAC	172
9.3.19	From TIM to IRTIM	172
10	Reset and clock control (RCC)	173
10.1	Reset	173
10.1.1	Power reset	173
10.1.2	System reset	173
10.1.3	RTC domain reset	174
10.2	Clocks	175
10.2.1	HSE clock	179
10.2.2	HSI clock	180
10.2.3	PLL	181
10.2.4	LSE clock	181
10.2.5	LSI clock	182
10.2.6	System clock (SYSCLK) selection	182

10.2.7	Clock security system (CSS)	182
10.2.8	ADC clock	183
10.2.9	RTC clock	183
10.2.10	Timers (TIMx) clock	183
10.2.11	Watchdog clock	184
10.2.12	I2S clock	184
10.2.13	Clock-out capability	184
10.2.14	Internal/external clock measurement with TIM16	185
10.3	Low-power modes	186
10.4	RCC registers	187
10.4.1	Clock control register (RCC_CR)	187
10.4.2	Clock configuration register (RCC_CFGR)	189
10.4.3	Clock interrupt register (RCC_CIR)	193
10.4.4	APB2 peripheral reset register (RCC_APB2RSTR)	195
10.4.5	APB1 peripheral reset register (RCC_APB1RSTR)	197
10.4.6	AHB peripheral clock enable register (RCC_AHBENR)	199
10.4.7	APB2 peripheral clock enable register (RCC_APB2ENR)	201
10.4.8	APB1 peripheral clock enable register (RCC_APB1ENR)	202
10.4.9	RTC domain control register (RCC_BDCR)	205
10.4.10	Control/status register (RCC_CSR)	207
10.4.11	AHB peripheral reset register (RCC_AHBSTR)	208
10.4.12	Clock configuration register 2 (RCC_CFGR2)	209
10.4.13	Clock configuration register 3 (RCC_CFGR3)	211
10.4.14	RCC register map	214
11	General-purpose I/Os (GPIO)	216
11.1	Introduction	216
11.2	GPIO main features	216
11.3	GPIO functional description	216
11.3.1	General-purpose I/O (GPIO)	218
11.3.2	I/O pin alternate function multiplexer and mapping	219
11.3.3	I/O port control registers	219
11.3.4	I/O port data registers	220
11.3.5	I/O data bitwise handling	220
11.3.6	GPIO locking mechanism	220
11.3.7	I/O alternate function input/output	221
11.3.8	External interrupt/wakeup lines	221

11.3.9	Input configuration	221
11.3.10	Output configuration	222
11.3.11	Alternate function configuration	222
11.3.12	Analog configuration	223
11.3.13	Using the HSE or LSE oscillator pins as GPIOs	224
11.3.14	Using the GPIO pins in the RTC supply domain	224
11.4	GPIO registers	225
11.4.1	GPIO port mode register (GPIOx_MODER) (x = A..H)	225
11.4.2	GPIO port output type register (GPIOx_OTYPER) (x = A..H)	225
11.4.3	GPIO port output speed register (GPIOx_OSPEEDR) (x = A..H)	226
11.4.4	GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..H)	226
11.4.5	GPIO port input data register (GPIOx_IDR) (x = A..H)	227
11.4.6	GPIO port output data register (GPIOx_ODR) (x = A..H)	227
11.4.7	GPIO port bit set/reset register (GPIOx_BSRR) (x = A..H)	227
11.4.8	GPIO port configuration lock register (GPIOx_LCKR)	228
11.4.9	GPIO alternate function low register (GPIOx_AFRL) (x = A..H)	229
11.4.10	GPIO alternate function high register (GPIOx_AFRH) (x = A..H)	229
11.4.11	GPIO port bit reset register (GPIOx_BRR) (x = A..H)	230
11.4.12	GPIO register map	231
12	System configuration controller (SYSCFG)	233
12.1	SYSCFG registers	233
12.1.1	SYSCFG configuration register 1 (SYSCFG_CFGR1)	233
12.1.2	SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)	235
12.1.3	SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)	236
12.1.4	SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)	238
12.1.5	SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)	240
12.1.6	SYSCFG configuration register 2 (SYSCFG_CFGR2)	241
12.1.7	SYSCFG register map	243
13	Direct memory access controller (DMA)	244

13.1	Introduction	244
13.2	DMA main features	244
13.3	DMA implementation	245
13.4	DMA functional description	246
13.4.1	DMA transactions	246
13.4.2	Arbiter	246
13.4.3	DMA channels	247
13.4.4	Programmable data width, data alignment and endians	248
13.4.5	Error management	249
13.4.6	DMA interrupts	250
13.4.7	DMA request mapping	251
13.5	DMA registers	258
13.5.1	DMA interrupt status register (DMA_ISR)	258
13.5.2	DMA interrupt flag clear register (DMA_IFCR)	259
13.5.3	DMA channel x configuration register (DMA_CCRx) (x = 1..7, where x = channel number)	260
13.5.4	DMA channel x number of data register (DMA_CNDTRx) (x = 1..7, where x = channel number)	262
13.5.5	DMA channel x peripheral address register (DMA_CPARx) (x = 1..7, where x = channel number)	262
13.5.6	DMA channel x memory address register (DMA_CMARx) (x = 1..7, where x = channel number)	263
13.5.7	DMA register map	264
14	Interrupts and events	266
14.1	Nested vectored interrupt controller (NVIC)	266
14.1.1	NVIC main features	266
14.1.2	SysTick calibration value register	266
14.1.3	Interrupt and exception vectors	266
14.2	Extended interrupts and events controller (EXTI)	273
14.2.1	Main features	273
14.2.2	Block diagram	274
14.2.3	Wakeup event management	274
14.2.4	Asynchronous Internal Interrupts	274
14.2.5	Functional description	275
14.2.6	External and internal interrupt/event line mapping	276
14.3	EXTI registers	278

14.3.1	Interrupt mask register (EXTI_IMR1)	278
14.3.2	Event mask register (EXTI_EMR1)	278
14.3.3	Rising trigger selection register (EXTI_RTSTR1)	279
14.3.4	Falling trigger selection register (EXTI_FTSTR1)	279
14.3.5	Software interrupt event register (EXTI_SWIER1)	280
14.3.6	Pending register (EXTI_PR1)	280
14.3.7	Interrupt mask register (EXTI_IMR2)	281
14.3.8	Event mask register (EXTI_EMR2)	281
14.3.9	Rising trigger selection register (EXTI_RTSTR2)	282
14.3.10	Falling trigger selection register (EXTI_FTSTR2)	282
14.3.11	Software interrupt event register (EXTI_SWIER2)	283
14.3.12	Pending register (EXTI_PR2)	283
14.3.13	EXTI register map	284
15	Analog-to-digital converters (ADC)	286
15.1	Introduction	286
15.2	ADC main features	287
15.3	ADC functional description	289
15.3.1	ADC block diagram	289
15.3.2	Pins and internal signals	290
15.3.3	Clocks	291
15.3.4	ADC1/2 connectivity	293
15.3.5	Slave AHB interface	294
15.3.6	ADC voltage regulator (ADVREGEN)	294
15.3.7	Single-ended and differential input channels	294
15.3.8	Calibration (ADCAL, ADCALDIF, ADCx_CALFACT)	295
15.3.9	ADC on-off control (ADEN, ADDIS, ADRDY)	298
15.3.10	Constraints when writing the ADC control bits	299
15.3.11	Channel selection (SQRx, JSQRx)	299
15.3.12	Channel-wise programmable sampling time (SMPR1, SMPR2)	300
15.3.13	Single conversion mode (CONT=0)	301
15.3.14	Continuous conversion mode (CONT=1)	301
15.3.15	Starting conversions (ADSTART, JADSTART)	302
15.3.16	Timing	303
15.3.17	Stopping an ongoing conversion (ADSTP, JADSTP)	303
15.3.18	Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)	305

15.3.19	Injected channel management	308
15.3.20	Discontinuous mode (DISCEN, DISCNUM, JDISCEN)	309
15.3.21	Queue of context for injected conversions	310
15.3.22	Programmable resolution (RES) - fast conversion mode	318
15.3.23	End of conversion, end of sampling phase (EOC, JEOC, EOSMP)	319
15.3.24	End of conversion sequence (EOS, JEOS)	319
15.3.25	Timing diagrams example (single/continuous modes, hardware/software triggers)	320
15.3.26	Data management	321
15.3.27	Dynamic low-power features	327
15.3.28	Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)	332
15.3.29	Dual ADC modes (STM32F302xB/C only)	336
15.3.30	Temperature sensor	349
15.3.31	VBAT supply monitoring	351
15.3.32	Monitoring the internal voltage reference	352
15.4	ADC interrupts	354
15.5	ADC registers (for each ADC)	355
15.5.1	ADC interrupt and status register (ADCx_ISR, x=1..2)	355
15.5.2	ADC interrupt enable register (ADCx_IER, x=1..2)	357
15.5.3	ADC control register (ADCx_CR, x=1..2)	359
15.5.4	ADC configuration register (ADCx_CFGR, x=1..2)	362
15.5.5	ADC sample time register 1 (ADCx_SMPR1, x=1..2)	366
15.5.6	ADC sample time register 2 (ADCx_SMPR2, x=1..2)	368
15.5.7	ADC watchdog threshold register 1 (ADCx_TR1, x=1..2)	368
15.5.8	ADC watchdog threshold register 2 (ADCx_TR2, x = 1..2)	369
15.5.9	ADC watchdog threshold register 3 (ADCx_TR3, x=1..2)	370
15.5.10	ADC regular sequence register 1 (ADCx_SQR1, x=1..2)	371
15.5.11	ADC regular sequence register 2 (ADCx_SQR2, x=1..2)	372
15.5.12	ADC regular sequence register 3 (ADCx_SQR3, x=1..2)	374
15.5.13	ADC regular sequence register 4 (ADCx_SQR4, x=1..2)	375
15.5.14	ADC regular Data Register (ADCx_DR, x=1..2)	376
15.5.15	ADC injected sequence register (ADCx_JSQR, x=1..2)	377
15.5.16	ADC offset register (ADCx_OFRy, x=1..2) (y=1..4)	379
15.5.17	ADC injected data register (ADCx_JDRy, x=1..2, y= 1..4)	380
15.5.18	ADC Analog Watchdog 2 Configuration Register (ADCx_AWD2CR, x=1..2)	380

15.5.19	ADC Analog Watchdog 3 Configuration Register (ADCx_AWD3CR, x=1..2)	381
15.5.20	ADC Differential Mode Selection Register (ADCx_DIFSEL, x=1..2) ..	381
15.5.21	ADC Calibration Factors (ADCx_CALFACT, x=1..2)	382
15.6	ADC common registers	383
15.6.1	ADC Common status register (ADCx_CSR, x=12)	383
15.6.2	ADC common control register (ADCx_CCR, x=12)	385
15.6.3	ADC common regular data register for dual mode (ADCx_CDR, x=12)	388
15.6.4	ADC register map	388
16	Digital-to-analog converter (DAC1)	392
16.1	Introduction	392
16.2	DAC1 main features	392
16.3	Single mode functional description	394
16.3.1	DAC channel enable	394
16.3.2	DAC output buffer enable	394
16.3.3	DAC data format	394
16.3.4	DAC channel conversion	394
16.3.5	DAC output voltage	396
16.3.6	DAC trigger selection	396
16.4	Noise generation	397
16.5	Triangle-wave generation	398
16.6	DMA request	399
16.7	DAC registers	399
16.7.1	DAC control register (DAC_CR)	400
16.7.2	DAC software trigger register (DAC_SWTRIGR)	402
16.7.3	DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)	402
16.7.4	DAC channel1 12-bit left-aligned data holding register (DAC_DHR12L1)	403
16.7.5	DAC channel1 8-bit right-aligned data holding register (DAC_DHR8R1)	403
16.7.6	DAC channel1 data output register (DAC_DOR1)	403
16.7.7	DAC status register (DAC_SR)	404
16.7.8	DAC register map	405
17	Comparator (COMP)	406

17.1	Introduction	406
17.2	COMP main features	406
17.3	COMP functional description	407
17.3.1	COMP block diagram	407
17.3.2	COMP pins and internal signals	407
17.3.3	COMP reset and clocks	409
17.3.4	Comparator LOCK mechanism	410
17.3.5	Hysteresis (STM32F302xBxC only)	410
17.3.6	Comparator output blanking function	410
17.3.7	Power mode (STM32F302xB/C only)	411
17.4	COMP interrupts	411
17.5	COMP registers	412
17.5.1	COMP1 control and status register (COMP1_CSR)	412
17.5.2	COMP2 control and status register (COMP2_CSR)	413
17.5.3	COMP4 control and status register (COMP4_CSR)	417
17.5.4	COMP6 control and status register (COMP6_CSR)	419
17.5.5	COMP register map	421
18	Operational amplifier (OPAMP)	422
18.1	OPAMP introduction	422
18.2	OPAMP main features	422
18.3	OPAMP functional description	422
18.3.1	General description	422
18.3.2	Clock	423
18.3.3	Operational amplifiers and comparators interconnections	423
18.3.4	Using the OPAMP outputs as ADC inputs	424
18.3.5	Calibration	424
18.3.6	Timer controlled Multiplexer mode	425
18.3.7	OPAMP modes	426
18.4	OPAMP registers	430
18.4.1	OPAMP1 control register (OPAMP1_CSR)	430
18.4.2	OPAMP2 control register (OPAMP2_CSR)	432
18.4.3	OPAMP register map	435
19	Touch sensing controller (TSC)	436
19.1	Introduction	436

19.2	TSC main features	436
19.3	TSC functional description	437
19.3.1	TSC block diagram	437
19.3.2	Surface charge transfer acquisition overview	437
19.3.3	Reset and clocks	439
19.3.4	Charge transfer acquisition sequence	440
19.3.5	Spread spectrum feature	441
19.3.6	Max count error	441
19.3.7	Sampling capacitor I/O and channel I/O mode selection	442
19.3.8	Acquisition mode	443
19.3.9	I/O hysteresis and analog switch control	443
19.3.10	Capacitive sensing GPIOs	444
19.4	TSC low-power modes	444
19.5	TSC interrupts	444
19.6	TSC registers	444
19.6.1	TSC control register (TSC_CR)	444
19.6.2	TSC interrupt enable register (TSC_IER)	446
19.6.3	TSC interrupt clear register (TSC_ICR)	447
19.6.4	TSC interrupt status register (TSC_ISR)	448
19.6.5	TSC I/O hysteresis control register (TSC_IOHCR)	448
19.6.6	TSC I/O analog switch control register (TSC_IOASCR)	449
19.6.7	TSC I/O sampling control register (TSC_IOSCR)	449
19.6.8	TSC I/O channel control register (TSC_IOCCR)	449
19.6.9	TSC I/O group control status register (TSC_I OGCSR)	450
19.6.10	TSC I/O group x counter register (TSC_I OGxCR) (x = 1..8)	450
19.6.11	TSC register map	451
20	Advanced-control timers (TIM1)	453
20.1	TIM1 introduction	453
20.2	TIM1 main features	453
20.3	TIM1 functional description	455
20.3.1	Time-base unit	455
20.3.2	Counter modes	457
20.3.3	Repetition counter	468
20.3.4	Clock selection	470
20.3.5	Capture/compare channels	474

20.3.6	Input capture mode	477
20.3.7	PWM input mode	478
20.3.8	Forced output mode	479
20.3.9	Output compare mode	479
20.3.10	PWM mode	481
20.3.11	Asymmetric PWM mode	484
20.3.12	Combined PWM mode	484
20.3.13	Combined 3-phase PWM mode	485
20.3.14	Complementary outputs and dead-time insertion	486
20.3.15	Using the break function	489
20.3.16	Clearing the OCxREF signal on an external event	495
20.3.17	6-step PWM generation	496
20.3.18	One-pulse mode	497
20.3.19	Retriggerable one pulse mode (OPM)	498
20.3.20	Encoder interface mode	499
20.3.21	UIF bit remapping	501
20.3.22	Timer input XOR function	502
20.3.23	Interfacing with Hall sensors	502
20.3.24	Timer synchronization	505
20.3.25	ADC synchronization	508
20.3.26	DMA burst mode	509
20.3.27	Debug mode	510
20.4	TIM1 registers	511
20.4.1	TIM1 control register 1 (TIMx_CR1)	511
20.4.2	TIM1 control register 2 (TIMx_CR2)	512
20.4.3	TIM1 slave mode control register (TIMx_SMCR)	515
20.4.4	TIM1 DMA/interrupt enable register (TIMx_DIER)	518
20.4.5	TIM1 status register (TIMx_SR)	519
20.4.6	TIM1 event generation register (TIMx_EGR)	521
20.4.7	TIM1 capture/compare mode register 1 (TIMx_CCMR1)	522
20.4.8	TIM1 capture/compare mode register 2 (TIMx_CCMR2)	527
20.4.9	TIM1 capture/compare enable register (TIMx_CCER)	528
20.4.10	TIM1 counter (TIMx_CNT)	532
20.4.11	TIM1 prescaler (TIMx_PSC)	532
20.4.12	TIM1 auto-reload register (TIMx_ARR)	532
20.4.13	TIM1 repetition counter register (TIMx_RCR)	533
20.4.14	TIM1 capture/compare register 1 (TIMx_CCR1)	533

20.4.15	TIM1 capture/compare register 2 (TIMx_CCR2)	534
20.4.16	TIM1 capture/compare register 3 (TIMx_CCR3)	534
20.4.17	TIM1 capture/compare register 4 (TIMx_CCR4)	535
20.4.18	TIM1 break and dead-time register (TIMx_BDTR)	535
20.4.19	TIM1 DMA control register (TIMx_DCR)	539
20.4.20	TIM1 DMA address for full transfer (TIMx_DMAR)	540
20.4.21	TIM1 capture/compare mode register 3 (TIMx_CCMR3)	540
20.4.22	TIM1 capture/compare register 5 (TIMx_CCR5)	541
20.4.23	TIM1 capture/compare register 6 (TIMx_CCR6)	542
20.4.24	TIM1 option registers (TIMx_OR)	542
20.4.25	TIM1 register map	543
21	General-purpose timers (TIM2/TIM3/TIM4)	546
21.1	TIM2/TIM3/TIM4 introduction	546
21.2	TIM2/TIM3/TIM4 main features	546
21.3	TIM2/TIM3/TIM4 functional description	548
21.3.1	Time-base unit	548
21.3.2	Counter modes	550
21.3.3	Clock selection	560
21.3.4	Capture/compare channels	564
21.3.5	Input capture mode	566
21.3.6	PWM input mode	568
21.3.7	Forced output mode	569
21.3.8	Output compare mode	569
21.3.9	PWM mode	570
21.3.10	Asymmetric PWM mode	574
21.3.11	Combined PWM mode	574
21.3.12	Clearing the OCxREF signal on an external event	575
21.3.13	One-pulse mode	577
21.3.14	Retriggerable one pulse mode (OPM)	578
21.3.15	Encoder interface mode	579
21.3.16	UIF bit remapping	581
21.3.17	Timer input XOR function	581
21.3.18	Timers and external trigger synchronization	582
21.3.19	Timer synchronization	585
21.3.20	DMA burst mode	590
21.3.21	Debug mode	591

21.4	TIM2/TIM3/TIM4 registers	592
21.4.1	TIMx control register 1 (TIMx_CR1)	592
21.4.2	TIMx control register 2 (TIMx_CR2)	594
21.4.3	TIMx slave mode control register (TIMx_SMCR)	595
21.4.4	TIMx DMA/Interrupt enable register (TIMx_DIER)	598
21.4.5	TIMx status register (TIMx_SR)	599
21.4.6	TIMx event generation register (TIMx_EGR)	601
21.4.7	TIMx capture/compare mode register 1 (TIMx_CCMR1)	602
21.4.8	TIMx capture/compare mode register 2 (TIMx_CCMR2)	606
21.4.9	TIMx capture/compare enable register (TIMx_CCER)	607
21.4.10	TIMx counter (TIMx_CNT)	609
21.4.11	TIMx prescaler (TIMx_PSC)	610
21.4.12	TIMx auto-reload register (TIMx_ARR)	610
21.4.13	TIMx capture/compare register 1 (TIMx_CCR1)	610
21.4.14	TIMx capture/compare register 2 (TIMx_CCR2)	611
21.4.15	TIMx capture/compare register 3 (TIMx_CCR3)	611
21.4.16	TIMx capture/compare register 4 (TIMx_CCR4)	612
21.4.17	TIMx DMA control register (TIMx_DCR)	613
21.4.18	TIMx DMA address for full transfer (TIMx_DMAR)	613
21.4.19	TIMx register map	614
22	General-purpose timers (TIM15/16/17)	616
22.1	TIM15/16/17 introduction	616
22.2	TIM15 main features	616
22.3	TIM16 and TIM17 main features	617
22.4	TIM15/16/17 functional description	620
22.4.1	Time-base unit	620
22.4.2	Counter modes	622
22.4.3	Repetition counter	626
22.4.4	Clock selection	627
22.4.5	Capture/compare channels	629
22.4.6	Input capture mode	632
22.4.7	PWM input mode (only for TIM15)	633
22.4.8	Forced output mode	634
22.4.9	Output compare mode	634
22.4.10	PWM mode	635
22.4.11	Combined PWM mode (TIM15 only)	636

22.4.12	Complementary outputs and dead-time insertion	638
22.4.13	Using the break function	640
22.4.14	One-pulse mode	643
22.4.15	UIF bit remapping	644
22.4.16	Timer input XOR function (TIM15 only)	645
22.4.17	External trigger synchronization (TIM15 only)	646
22.4.18	Slave mode: Combined reset + trigger mode	648
22.4.19	DMA burst mode	648
22.4.20	Debug mode	649
22.5	TIM15 registers	650
22.5.1	TIM15 control register 1 (TIM15_CR1)	650
22.5.2	TIM15 control register 2 (TIM15_CR2)	651
22.5.3	TIM15 slave mode control register (TIM15_SMCR)	653
22.5.4	TIM15 DMA/interrupt enable register (TIM15_DIER)	655
22.5.5	TIM15 status register (TIM15_SR)	656
22.5.6	TIM15 event generation register (TIM15_EGR)	657
22.5.7	TIM15 capture/compare mode register 1 (TIM15_CCMR1)	658
22.5.8	TIM15 capture/compare enable register (TIM15_CCER)	662
22.5.9	TIM15 counter (TIM15_CNT)	665
22.5.10	TIM15 prescaler (TIM15_PSC)	666
22.5.11	TIM15 auto-reload register (TIM15_ARR)	666
22.5.12	TIM15 repetition counter register (TIM15_RCR)	667
22.5.13	TIM15 capture/compare register 1 (TIM15_CCR1)	667
22.5.14	TIM15 capture/compare register 2 (TIM15_CCR2)	668
22.5.15	TIM15 break and dead-time register (TIM15_BDTR)	668
22.5.16	TIM15 DMA control register (TIM15_DCR)	671
22.5.17	TIM15 DMA address for full transfer (TIM15_DMAR)	671
22.5.18	TIM15 register map	672
22.6	TIM16&TIM17 registers	674
22.6.1	TIM16&TIM17 control register 1 (TIMx_CR1)	674
22.6.2	TIM16&TIM17 control register 2 (TIMx_CR2)	675
22.6.3	TIM16 &TIM17 DMA/interrupt enable register (TIMx_DIER)	676
22.6.4	TIM16&TIM17 status register (TIMx_SR)	678
22.6.5	TIM16&TIM17 event generation register (TIMx_EGR)	679
22.6.6	TIM16&TIM17 capture/compare mode register 1 (TIMx_CCMR1)	680
22.6.7	TIM16&TIM17 capture/compare enable register (TIMx_CCER)	682
22.6.8	TIM16&TIM17 counter (TIMx_CNT)	684

	22.6.9	TIM16&TIM17 prescaler (TIMx_PSC)	685
	22.6.10	TIM16&TIM17 auto-reload register (TIMx_ARR)	685
	22.6.11	TIM16&TIM17 repetition counter register (TIMx_RCR)	686
	22.6.12	TIM16&TIM17 capture/compare register 1 (TIMx_CCR1)	686
	22.6.13	TIM16&TIM17 break and dead-time register (TIMx_BDTR)	687
	22.6.14	TIM16&TIM17 DMA control register (TIMx_DCR)	689
	22.6.15	TIM16&TIM17 DMA address for full transfer (TIMx_DMAR)	690
	22.6.16	TIM16 option register (TIM16_OR)	690
	22.6.17	TIM16&TIM17 register map	692
23		Basic timers (TIM6)	694
	23.1	TIM6 introduction	694
	23.2	TIM6 main features	694
	23.3	TIM6 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 registers	701
	23.4.1	TIM6 control register 1 (TIMx_CR1)	701
	23.4.2	TIM6 control register 2 (TIMx_CR2)	703
	23.4.3	TIM6 DMA/Interrupt enable register (TIMx_DIER)	703
	23.4.4	TIM6 status register (TIMx_SR)	704
	23.4.5	TIM6 event generation register (TIMx_EGR)	704
	23.4.6	TIM6 counter (TIMx_CNT)	704
	23.4.7	TIM6 prescaler (TIMx_PSC)	705
	23.4.8	TIM6 auto-reload register (TIMx_ARR)	705
	23.4.9	TIM6 register map	706
24		Infrared interface (IRTIM)	707
25		System window watchdog (WWDG)	708
	25.1	Introduction	708
	25.2	WWDG main features	708
	25.3	WWDG functional description	708
	25.3.1	Enabling the watchdog	709

25.3.2	Controlling the downcounter	709
25.3.3	Advanced watchdog interrupt feature	709
25.3.4	How to program the watchdog timeout	710
25.3.5	Debug mode	711
25.4	WWDG registers	711
25.4.1	Control register (WWDG_CR)	711
25.4.2	Configuration register (WWDG_CFR)	712
25.4.3	Status register (WWDG_SR)	712
25.4.4	WWDG register map	713
26	Independent watchdog (IWDG)	714
26.1	Introduction	714
26.2	IWDG main features	714
26.3	IWDG functional description	714
26.3.1	IWDG block diagram	714
26.3.2	Window option	715
26.3.3	Hardware watchdog	715
26.3.4	Register access protection	716
26.3.5	Debug mode	716
26.4	IWDG registers	716
26.4.1	Key register (IWDG_KR)	716
26.4.2	Prescaler register (IWDG_PR)	717
26.4.3	Reload register (IWDG_RLR)	718
26.4.4	Status register (IWDG_SR)	719
26.4.5	Window register (IWDG_WINR)	720
26.4.6	IWDG register map	721
27	Real-time clock (RTC)	722
27.1	Introduction	722
27.2	RTC main features	723
27.3	RTC functional description	724
27.3.1	RTC block diagram	724
27.3.2	GPIOs controlled by the RTC	725
27.3.3	Clock and prescalers	727
27.3.4	Real-time clock and calendar	727
27.3.5	Programmable alarms	728

27.3.6	Periodic auto-wakeup	728
27.3.7	RTC initialization and configuration	729
27.3.8	Reading the calendar	730
27.3.9	Resetting the RTC	731
27.3.10	RTC synchronization	732
27.3.11	RTC reference clock detection	732
27.3.12	RTC smooth digital calibration	733
27.3.13	Time-stamp function	735
27.3.14	Tamper detection	736
27.3.15	Calibration clock output	737
27.3.16	Alarm output	738
27.4	RTC low-power modes	738
27.5	RTC interrupts	738
27.6	RTC registers	739
27.6.1	RTC time register (RTC_TR)	740
27.6.2	RTC date register (RTC_DR)	741
27.6.3	RTC control register (RTC_CR)	742
27.6.4	RTC initialization and status register (RTC_ISR)	745
27.6.5	RTC prescaler register (RTC_PRER)	748
27.6.6	RTC wakeup timer register (RTC_WUTR)	749
27.6.7	RTC alarm A register (RTC_ALRMAR)	750
27.6.8	RTC alarm B register (RTC_ALRMBR)	751
27.6.9	RTC write protection register (RTC_WPR)	752
27.6.10	RTC sub second register (RTC_SSR)	753
27.6.11	RTC shift control register (RTC_SHIFTR)	754
27.6.12	RTC timestamp time register (RTC_TSTR)	755
27.6.13	RTC timestamp date register (RTC_TSDR)	756
27.6.14	RTC time-stamp sub second register (RTC_TSSSR)	757
27.6.15	RTC calibration register (RTC_CALR)	758
27.6.16	RTC tamper and alternate function configuration register (RTC_TAFCR)	759
27.6.17	RTC alarm A sub second register (RTC_ALRMASR)	762
27.6.18	RTC alarm B sub second register (RTC_ALRMBSSR)	763
27.6.19	RTC backup registers (RTC_BKPxR)	764
27.6.20	RTC register map	765
28	Inter-integrated circuit (I2C) interface	767

28.1	Introduction	767
28.2	I2C main features	767
28.3	I2C implementation	768
28.4	I2C functional description	768
28.4.1	I2C block diagram	769
28.4.2	I2C clock requirements	770
28.4.3	Mode selection	770
28.4.4	I2C initialization	772
28.4.5	Software reset	776
28.4.6	Data transfer	777
28.4.7	I2C slave mode	779
28.4.8	I2C master mode	788
28.4.9	I2C_TIMINGR register configuration examples	800
28.4.10	SMBus specific features	801
28.4.11	SMBus initialization	804
28.4.12	SMBus: I2C_TIMEOCTR register configuration examples	806
28.4.13	SMBus slave mode	807
28.4.14	Wakeup from Stop mode on address match	814
28.4.15	Error conditions	814
28.4.16	DMA requests	816
28.4.17	Debug mode	817
28.5	I2C low-power modes	817
28.6	I2C interrupts	817
28.7	I2C registers	819
28.7.1	Control register 1 (I2C_CR1)	819
28.7.2	Control register 2 (I2C_CR2)	822
28.7.3	Own address 1 register (I2C_OAR1)	825
28.7.4	Own address 2 register (I2C_OAR2)	826
28.7.5	Timing register (I2C_TIMINGR)	827
28.7.6	Timeout register (I2C_TIMEOCTR)	828
28.7.7	Interrupt and status register (I2C_ISR)	829
28.7.8	Interrupt clear register (I2C_ICR)	831
28.7.9	PEC register (I2C_PECR)	832
28.7.10	Receive data register (I2C_RXDR)	833
28.7.11	Transmit data register (I2C_TXDR)	833
28.7.12	I2C register map	834

29	Universal synchronous asynchronous receiver transmitter (USART)	836
29.1	Introduction	836
29.2	USART main features	836
29.3	USART extended features	837
29.4	USART implementation	838
29.5	USART functional description	838
29.5.1	USART character description	841
29.5.2	Transmitter	842
29.5.3	Receiver	845
29.5.4	Baud rate generation	852
29.5.5	Tolerance of the USART receiver to clock deviation	854
29.5.6	Auto baud rate detection	855
29.5.7	Multiprocessor communication	856
29.5.8	Modbus communication	858
29.5.9	Parity control	859
29.5.10	LIN (local interconnection network) mode	860
29.5.11	USART synchronous mode	862
29.5.12	Single-wire half-duplex communication	865
29.5.13	Smartcard mode	865
29.5.14	IrDA SIR ENDEC block	870
29.5.15	Continuous communication using DMA	872
29.5.16	RS232 Hardware flow control and RS485 Driver Enable	874
29.5.17	Wakeup from Stop mode	876
29.6	USART low-power modes	877
29.7	USART interrupts	877
29.8	USART registers	879
29.8.1	Control register 1 (USARTx_CR1)	879
29.8.2	Control register 2 (USARTx_CR2)	882
29.8.3	Control register 3 (USARTx_CR3)	886
29.8.4	Baud rate register (USARTx_BRR)	890
29.8.5	Guard time and prescaler register (USARTx_GTPR)	890
29.8.6	Receiver timeout register (USARTx_RTOR)	891
29.8.7	Request register (USARTx_RQR)	892
29.8.8	Interrupt & status register (USARTx_ISR)	893
29.8.9	Interrupt flag clear register (USARTx_ICR)	898

	29.8.10	Receive data register (USARTx_RDR)	899
	29.8.11	Transmit data register (USARTx_TDR)	899
	29.8.12	USART register map	900
30		Serial peripheral interface / inter-IC sound (SPI/I2S)	902
	30.1	Introduction	902
	30.2	SPI main features	902
	30.3	I2S main features	903
	30.4	SPI/I2S implementation	903
	30.5	SPI functional description	904
	30.5.1	General description	904
	30.5.2	Communications between one master and one slave	905
	30.5.3	Standard multi-slave communication	907
	30.5.4	Slave select (NSS) pin management	908
	30.5.5	Communication formats	910
	30.5.6	Configuration of SPI	912
	30.5.7	Procedure for enabling SPI	913
	30.5.8	Data transmission and reception procedures	913
	30.5.9	SPI status flags	923
	30.5.10	SPI error flags	924
	30.5.11	NSS pulse mode	925
	30.5.12	TI mode	925
	30.5.13	CRC calculation	926
	30.6	SPI interrupts	928
	30.7	I ² S functional description	929
	30.7.1	I ² S general description	929
	30.7.2	I2S full duplex	930
	30.7.3	Supported audio protocols	931
	30.7.4	Clock generator	938
	30.7.5	I ² S master mode	940
	30.7.6	I ² S slave mode	942
	30.7.7	I ² S status flags	944
	30.7.8	I ² S error flags	945
	30.7.9	DMA features	946
	30.8	I ² S interrupts	946
	30.9	SPI and I ² S registers	947

30.9.1	SPI control register 1 (SPIx_CR1)	947
30.9.2	SPI control register 2 (SPIx_CR2)	949
30.9.3	SPI status register (SPIx_SR)	952
30.9.4	SPI data register (SPIx_DR)	954
30.9.5	SPI CRC polynomial register (SPIx_CRCPR)	954
30.9.6	SPI Rx CRC register (SPIx_RXCRCR)	955
30.9.7	SPI Tx CRC register (SPIx_TXCRCR)	955
30.9.8	SPIx_I ² S configuration register (SPIx_I2SCFGR)	956
30.9.9	SPIx_I ² S prescaler register (SPIx_I2SPR)	958
30.9.10	SPI/I2S register map	959
31	Controller area network (bxCAN)	960
31.1	Introduction	960
31.2	bxCAN main features	960
31.3	bxCAN general description	961
31.3.1	CAN 2.0B active core	961
31.3.2	Control, status and configuration registers	961
31.3.3	Tx mailboxes	961
31.3.4	Acceptance filters	962
31.4	bxCAN operating modes	962
31.4.1	Initialization mode	962
31.4.2	Normal mode	962
31.4.3	Sleep mode (low-power)	963
31.5	Test mode	964
31.5.1	Silent mode	964
31.5.2	Loop back mode	965
31.5.3	Loop back combined with silent mode	965
31.6	Behavior in Debug mode	966
31.7	bxCAN functional description	966
31.7.1	Transmission handling	966
31.7.2	Time triggered communication mode	968
31.7.3	Reception handling	968
31.7.4	Identifier filtering	969
31.7.5	Message storage	973
31.7.6	Error management	975
31.7.7	Bit timing	975

31.8	bxCAN interrupts	978
31.9	CAN registers	979
31.9.1	Register access protection	979
31.9.2	CAN control and status registers	979
31.9.3	CAN mailbox registers	989
31.9.4	CAN filter registers	997
31.9.5	bxCAN register map	1001
32	Universal serial bus full-speed device interface (USB)	1006
32.1	Introduction	1006
32.2	USB main features	1006
32.3	USB implementation	1006
32.4	USB functional description	1007
32.4.1	Description of USB blocks	1008
32.5	Programming considerations	1009
32.5.1	Generic USB device programming	1009
32.5.2	System and power-on reset	1010
32.5.3	Double-buffered endpoints	1014
32.5.4	Isochronous transfers	1017
32.5.5	Suspend/Resume events	1018
32.6	USB registers	1020
32.6.1	Common registers	1020
32.6.2	Endpoint-specific registers	1028
32.6.3	Buffer descriptor table	1033
32.6.4	USB register map	1037
33	Debug support (DBG)	1039
33.1	Overview	1039
33.2	Reference ARM® documentation	1040
33.3	SWJ debug port (serial wire and JTAG)	1040
33.3.1	Mechanism to select the JTAG-DP or the SW-DP	1041
33.4	Pinout and debug port pins	1041
33.4.1	SWJ debug port pins	1042
33.4.2	Flexible SWJ-DP pin assignment	1042
33.4.3	Internal pull-up and pull-down on JTAG pins	1043
33.4.4	Using serial wire and releasing the unused debug pins as GPIOs	1044

33.5	STM32F302xx JTAG TAP connection	1044
33.6	ID codes and locking mechanism	1045
33.6.1	MCU device ID code	1046
33.6.2	Boundary scan TAP	1046
33.6.3	Cortex-M4 [®] F TAP	1046
33.6.4	Cortex-M4 [®] F JEDEC-106 ID code	1047
33.7	JTAG debug port	1047
33.8	SW debug port	1049
33.8.1	SW protocol introduction	1049
33.8.2	SW protocol sequence	1049
33.8.3	SW-DP state machine (reset, idle states, ID code)	1050
33.8.4	DP and AP read/write accesses	1050
33.8.5	SW-DP registers	1051
33.8.6	SW-AP registers	1052
33.9	AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP	1052
33.10	Core debug	1053
33.11	Capability of the debugger host to connect under system reset	1053
33.12	FPB (Flash patch breakpoint)	1054
33.13	DWT (data watchpoint trigger)	1055
33.14	MCU debug component (DBGMCU)	1055
33.14.1	Debug support for low-power modes	1055
33.14.2	Debug support for timers, watchdog, bxCAN and I ² C	1056
33.14.3	Debug MCU configuration register	1056
33.14.4	Debug MCU APB1 freeze register (DBGMCU_APB1_FZ)	1058
33.14.5	Debug MCU APB2 freeze register (DBGMCU_APB2_FZ)	1060
33.15	TPIU (trace port interface unit)	1060
33.15.1	Introduction	1060
33.15.2	TRACE pin assignment	1061
33.15.3	TPUI formatter	1063
33.15.4	TPUI frame synchronization packets	1063
33.15.5	Transmission of the synchronization frame packet	1064
33.15.6	Synchronous mode	1064
33.15.7	Asynchronous mode	1064
33.15.8	TRACECLKIN connection inside the STM32F302xx	1065
33.15.9	TPIU registers	1066

	33.15.10 Example of configuration	1067
	33.16 DBG register map	1068
34	Device electronic signature	1069
	34.1 Unique device ID register (96 bits)	1069
	34.2 Memory size data register	1070
	34.2.1 Flash size data register	1070
35	Revision history	1071

List of tables

Table 1.	Available features related to each product	40
Table 2.	STM32F302xB/C peripheral register boundary addresses.	48
Table 3.	STM32F302xD/E peripheral register boundary addresses.	51
Table 4.	STM32F302x6/x8 peripheral register boundary addresses	54
Table 5.	Boot modes.	57
Table 6.	Flash module organization	59
Table 7.	Flash memory read protection status	68
Table 8.	Access status versus protection level and execution modes	70
Table 9.	Flash interrupt request	71
Table 10.	Flash interface - register map and reset values	77
Table 11.	Option byte format	79
Table 12.	Option byte organization.	79
Table 13.	Description of the option bytes	80
Table 14.	CRC register map and reset values	87
Table 15.	NOR/PSRAM bank selection	92
Table 16.	NOR/PSRAM External memory address	92
Table 17.	NAND/PC Card memory mapping and timing registers	93
Table 18.	NAND bank selection	93
Table 19.	Programmable NOR/PSRAM access parameters	95
Table 20.	Non-multiplexed I/O NOR Flash memory	95
Table 21.	16-bit multiplexed I/O NOR Flash memory	96
Table 22.	Non-multiplexed I/Os PSRAM/SRAM	96
Table 23.	16-Bit multiplexed I/O PSRAM	96
Table 24.	NOR Flash/PSRAM: Example of supported memories and transactions	97
Table 25.	FMC_BCRx bit fields.	100
Table 26.	FMC_BTRx bit fields.	101
Table 27.	FMC_BCRx bit fields.	103
Table 28.	FMC_BTRx bit fields.	103
Table 29.	FMC_BWTRx bit fields	104
Table 30.	FMC_BCRx bit fields.	106
Table 31.	FMC_BTRx bit fields.	106
Table 32.	FMC_BWTRx bit fields	107
Table 33.	FMC_BCRx bit fields.	108
Table 34.	FMC_BTRx bit fields.	109
Table 35.	FMC_BWTRx bit fields	109
Table 36.	FMC_BCRx bit fields.	111
Table 37.	FMC_BTRx bit fields.	111
Table 38.	FMC_BWTRx bit fields	112
Table 39.	FMC_BCRx bit fields.	113
Table 40.	FMC_BTRx bit fields.	114
Table 41.	FMC_BCRx bit fields.	119
Table 42.	FMC_BTRx bit fields.	120
Table 43.	FMC_BCRx bit fields.	121
Table 44.	FMC_BTRx bit fields.	122
Table 45.	Programmable NAND Flash/PC Card access parameters	130
Table 46.	8-bit NAND Flash	130
Table 47.	16-bit NAND Flash	131
Table 48.	16-bit PC Card	131

Table 49.	Supported memories and transactions	132
Table 50.	16-bit PC-Card signals and access type	137
Table 51.	ECC result relevant bits	144
Table 52.	FMC register map	145
Table 53.	Low-power mode summary	152
Table 54.	Sleep-now	154
Table 55.	Sleep-on-exit	154
Table 56.	Stop mode	156
Table 57.	Standby mode	157
Table 58.	PWR register map and reset values	162
Table 59.	STM32F302xx peripherals interconnect matrix	163
Table 60.	VREFOPAMPx to ADC channel	167
Table 61.	OPAMP output to ADC input	167
Table 62.	Comparator outputs to timer inputs	168
Table 63.	Timer output selection as comparator blanking source	169
Table 64.	Timer synchronization	171
Table 65.	Timer and EXTI signals triggering DAC1 conversions	172
Table 66.	RCC register map and reset values	214
Table 67.	Port bit configuration table	218
Table 68.	GPIO register map and reset values	231
Table 69.	SYSCFG register map and reset values	243
Table 70.	DMA implementation	245
Table 71.	Programmable data width & endian behavior (when bits PINC = MINC = 1)	248
Table 72.	DMA interrupt requests	250
Table 73.	STM32F302xB/C/D/E summary of DMA1 requests for each channel	254
Table 74.	STM32F302x6/8 summary of DMA1 requests for each channel	254
Table 75.	STM32F302xB/C/D/E summary of DMA2 requests for each channel	257
Table 76.	DMA register map and reset values	264
Table 77.	STM32F302xB/C/D/E vector table	266
Table 78.	STM32F302x6/8 vector table	269
Table 79.	External interrupt/event controller register map and reset values	284
Table 80.	ADC external channels mapping	287
Table 81.	ADC internal channels summary	288
Table 82.	ADC internal signals	290
Table 83.	ADC pins	290
Table 84.	Configuring the trigger polarity for regular external triggers	305
Table 85.	Configuring the trigger polarity for injected external triggers	306
Table 86.	ADC1 (master) & 2 (slave) - External triggers for regular channels	307
Table 87.	ADC1 & ADC2 - External trigger for injected channels	307
Table 88.	TSAR timings depending on resolution	319
Table 89.	Offset computation versus data resolution	322
Table 90.	Analog watchdog channel selection	332
Table 91.	Analog watchdog 1 comparison	333
Table 92.	Analog watchdog 2 and 3 comparison	333
Table 93.	ADC interrupts per each ADC	354
Table 94.	DELAY bits versus ADC resolution	387
Table 95.	ADC global register map	388
Table 96.	ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC, x=1..2)	389
Table 97.	ADC register map and reset values (master and slave ADC common registers) offset =0x300, x=1 or 34)	391
Table 98.	DAC1 pins	393

Table 99.	External triggers (DAC1)	396
Table 100.	External triggers (DAC1)	397
Table 101.	DAC register map and reset values	405
Table 102.	STM32F302xB/C/D/E comparator input/outputs summary	408
Table 103.	STM32F302x6/8 comparator input/outputs summary	409
Table 104.	COMP register map and reset values	421
Table 105.	Connections with dedicated I/O	422
Table 106.	OPAMP register map and reset values	435
Table 107.	Acquisition sequence summary	439
Table 108.	Spread spectrum deviation versus AHB clock frequency	441
Table 109.	I/O state depending on its mode and IODEF bit value	442
Table 110.	Effect of low-power modes on TSC	444
Table 111.	Interrupt control bits	444
Table 112.	TSC register map and reset values	451
Table 113.	Behavior of timer outputs versus BRK/BRK2 inputs	493
Table 114.	Counting direction versus encoder signals	500
Table 115.	TIMx internal trigger connection	517
Table 116.	Output control bits for complementary OCx and OCxN channels with break feature	531
Table 117.	TIM1 register map and reset values	543
Table 118.	Counting direction versus encoder signals	580
Table 119.	TIMx internal trigger connection	597
Table 120.	Output control bit for standard OCx channels	609
Table 121.	TIM2/TIM3/TIM4 register map and reset values	614
Table 122.	TIMx Internal trigger connection	654
Table 123.	Output control bits for complementary OCx and OCxN channels with break feature	665
Table 124.	TIM15 register map and reset values	672
Table 125.	Output control bits for complementary OCx and OCxN channels with break feature	684
Table 126.	TIM16&TIM17 register map and reset values	692
Table 127.	TIM6 register map and reset values	706
Table 128.	WWDG register map and reset values	713
Table 129.	IWDG register map and reset values	721
Table 130.	RTC pin PC13 configuration	725
Table 131.	LSE pin PC14 configuration	726
Table 132.	LSE pin PC15 configuration	726
Table 133.	Effect of low-power modes on RTC	738
Table 134.	Interrupt control bits	739
Table 135.	RTC register map and reset values	765
Table 136.	STM32F302xx I2C implementation	768
Table 137.	Comparison of analog vs. digital filters	772
Table 138.	I2C-SMBUS specification data setup and hold times	775
Table 139.	I2C configuration table	779
Table 140.	I2C-SMBUS specification clock timings	789
Table 141.	Examples of timings settings for fI2CCLK = 8 MHz	800
Table 142.	Examples of timings settings for fI2CCLK = 16 MHz	800
Table 143.	Examples of timings settings for fI2CCLK = 48 MHz	801
Table 144.	SMBus timeout specifications	803
Table 145.	SMBUS with PEC configuration table	805
Table 146.	Examples of TIMEOUTA settings for various I2CCLK frequencies (max t _{TIMEOUT} = 25 ms)	806
Table 147.	Examples of TIMEOUTB settings for various I2CCLK frequencies	807
Table 148.	Examples of TIMEOUTA settings for various I2CCLK frequencies (max t _{IDLE} = 50 µs)	807

Table 149.	low-power modes	817
Table 150.	I2C Interrupt requests	817
Table 151.	I2C register map and reset values	834
Table 152.	STM32F302xx USART features	838
Table 153.	Noise detection from sampled data	850
Table 154.	Error calculation for programmed baud rates at $f_{CK} = 72\text{MHz}$ in both cases of oversampling by 16 or by 8.	853
Table 155.	Tolerance of the USART receiver when BRR [3:0] = 0000.	855
Table 156.	Tolerance of the USART receiver when BRR[3:0] is different from 0000.	855
Table 157.	Frame formats.	859
Table 158.	Effect of low-power modes on the USART	877
Table 159.	USART interrupt requests.	877
Table 160.	USART register map and reset values	900
Table 161.	STM32F302x6/8 SPI implementation	903
Table 162.	STM32F302xB/C/D/E SPI implementation	904
Table 163.	SPI interrupt requests	928
Table 164.	Audio-frequency precision using standard 8 MHz HSE	939
Table 165.	I ² S interrupt requests	946
Table 166.	SPI register map and reset values	959
Table 167.	Transmit mailbox mapping	974
Table 168.	Receive mailbox mapping.	974
Table 169.	bxCAN register map and reset values	1001
Table 170.	STM32F302xx USB implementation.	1006
Table 171.	Double-buffering buffer flag definition.	1015
Table 172.	Bulk double-buffering memory buffers usage	1016
Table 173.	Isochronous memory buffers usage	1017
Table 174.	Resume event detection	1019
Table 175.	Reception status encoding	1031
Table 176.	Endpoint type encoding	1032
Table 177.	Endpoint kind meaning	1032
Table 178.	Transmission status encoding	1032
Table 179.	Definition of allocated buffer memory	1036
Table 180.	USB register map and reset values	1037
Table 181.	SWJ debug port pins.	1042
Table 182.	Flexible SWJ-DP pin assignment	1042
Table 183.	JTAG debug port data registers	1047
Table 184.	32-bit debug port registers addressed through the shifted value A[3:2]	1048
Table 185.	Packet request (8-bits)	1049
Table 186.	ACK response (3 bits).	1050
Table 187.	DATA transfer (33 bits)	1050
Table 188.	SW-DP registers	1051
Table 189.	Cortex-M4 [®] F AHB-AP registers	1052
Table 190.	Core debug registers	1053
Table 191.	Asynchronous TRACE pin assignment.	1061
Table 192.	Synchronous TRACE pin assignment.	1061
Table 193.	Flexible TRACE pin assignment	1062
Table 194.	Important TPIU registers.	1066
Table 195.	DBG register map and reset values	1068
Table 196.	Document revision history.	1071

List of figures

Figure 1.	STM32F302xB/C system architecture	45
Figure 2.	STM32F302x6/8 system architecture	45
Figure 3.	STM32F302xD/E system architecture	46
Figure 4.	Programming procedure	63
Figure 5.	Flash memory Page Erase procedure	65
Figure 6.	Flash memory Mass Erase procedure	66
Figure 7.	CRC calculation unit block diagram	83
Figure 8.	FMC block diagram	89
Figure 9.	FMC memory banks	92
Figure 10.	Mode1 read access waveforms	99
Figure 11.	Mode1 write access waveforms	100
Figure 12.	ModeA read access waveforms	102
Figure 13.	ModeA write access waveforms	102
Figure 14.	Mode2 and mode B read access waveforms	104
Figure 15.	Mode2 write access waveforms	105
Figure 16.	ModeB write access waveforms	105
Figure 17.	ModeC read access waveforms	107
Figure 18.	ModeC write access waveforms	108
Figure 19.	ModeD read access waveforms	110
Figure 20.	ModeD write access waveforms	110
Figure 21.	Muxed read access waveforms	112
Figure 22.	Muxed write access waveforms	113
Figure 23.	Asynchronous wait during a read access waveforms	115
Figure 24.	Asynchronous wait during a write access waveforms	116
Figure 25.	Wait configuration waveforms	118
Figure 26.	Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)	119
Figure 27.	Synchronous multiplexed write mode waveforms - PSRAM (CRAM)	121
Figure 28.	NAND Flash/PC Card controller waveforms for common memory access	133
Figure 29.	Access to non 'CE don't care' NAND-Flash	134
Figure 30.	Power supply overview	147
Figure 31.	Power on reset/power down reset waveform	150
Figure 32.	PVD thresholds	151
Figure 33.	Simplified diagram of the reset circuit	174
Figure 34.	STM32F302xB/C clock tree	176
Figure 35.	STM32F302xD/E clock tree	177
Figure 36.	STM32F302x6/8 clock tree	178
Figure 37.	HSE/ LSE clock sources	179
Figure 38.	Frequency measurement with TIM16 in capture mode	185
Figure 39.	Basic structure of an I/O port bit	217
Figure 40.	Basic structure of a five-volt tolerant I/O port bit	217
Figure 41.	Input floating/pull up/pull down configurations	221
Figure 42.	Output configuration	222
Figure 43.	Alternate function configuration	223
Figure 44.	High impedance-analog configuration	223
Figure 45.	DMA block diagram	245
Figure 46.	STM32F302xB/C/D/E and STM32F302x6/8 DMA1 request mapping	252
Figure 47.	STM32F302x6/8 DMA1 request mapping	253
Figure 48.	STM32F302xB/C DMA2 request mapping	256

Figure 49.	External interrupt/event block diagram	274
Figure 50.	External interrupt/event GPIO mapping	276
Figure 51.	ADC block diagram	289
Figure 52.	ADC clock scheme	291
Figure 53.	ADC1 and ADC2 connectivity	293
Figure 54.	ADC calibration	296
Figure 55.	Updating the ADC calibration factor	297
Figure 56.	Mixing single-ended and differential channels	297
Figure 57.	Enabling / Disabling the ADC	298
Figure 58.	Analog to digital conversion time	303
Figure 59.	Stopping ongoing regular conversions	304
Figure 60.	Stopping ongoing regular and injected conversions	305
Figure 61.	Triggers are shared between ADC master & ADC slave	306
Figure 62.	Injected conversion latency.	309
Figure 63.	Example of JSQR queue of context (sequence change)	312
Figure 64.	Example of JSQR queue of context (trigger change)	312
Figure 65.	Example of JSQR queue of context with overflow before conversion	313
Figure 66.	Example of JSQR queue of context with overflow during conversion	313
Figure 67.	Example of JSQR queue of context with empty queue (case JQM=0).	314
Figure 68.	Example of JSQR queue of context with empty queue (case JQM=1).	314
Figure 69.	Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion.	315
Figure 70.	Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.	315
Figure 71.	Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs outside an ongoing conversion	316
Figure 72.	Flushing JSQR queue of context by setting JADSTP=1 (JQM=1)	316
Figure 73.	Flushing JSQR queue of context by setting ADDIS=1 (JQM=0).	317
Figure 74.	Flushing JSQR queue of context by setting ADDIS=1 (JQM=1).	317
Figure 75.	Example of JSQR queue of context when changing SW and HW triggers.	318
Figure 76.	Single conversions of a sequence, software trigger	320
Figure 77.	Continuous conversion of a sequence, software trigger	320
Figure 78.	Single conversions of a sequence, hardware trigger	321
Figure 79.	Continuous conversions of a sequence, hardware trigger	321
Figure 80.	Right alignment (offset disabled, unsigned value)	323
Figure 81.	Right alignment (offset enabled, signed value).	323
Figure 82.	Left alignment (offset disabled, unsigned value).	324
Figure 83.	Left alignment (offset enabled, signed value).	324
Figure 84.	Example of overrun (OVR)	325
Figure 85.	AUTODLY=1, regular conversion in continuous mode, software trigger	328
Figure 86.	AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0)	329
Figure 87.	AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1)	330
Figure 88.	AUTODLY=1, regular continuous conversions interrupted by injected conversions	331
Figure 89.	AUTODLY=1 in auto- injected mode (JAUTO=1).	331
Figure 90.	Analog watchdog's guarded area	332
Figure 91.	ADCy_AWDx_OUT signal generation (on all regular channels).	334
Figure 92.	ADCy_AWDx_OUT signal generation (AWDx flag not cleared by SW)	335
Figure 93.	ADCy_AWDx_OUT signal generation (on a single regular channel)	335
Figure 94.	ADCy_AWDx_OUT signal generation (on all injected channels)	335

Figure 95.	Dual ADC block diagram ⁽¹⁾	337
Figure 96.	Injected simultaneous mode on 4 channels: dual ADC mode	338
Figure 97.	Regular simultaneous mode on 16 channels: dual ADC mode	340
Figure 98.	Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode.	342
Figure 99.	Interleaved mode on 1 channel in single conversion mode: dual ADC mode.	342
Figure 100.	Interleaved conversion with injection	343
Figure 101.	Alternate trigger: injected group of each ADC	344
Figure 102.	Alternate trigger: 4 injected channels (each ADC) in discontinuous mode.	345
Figure 103.	Alternate + regular simultaneous	346
Figure 104.	Case of trigger occurring during injected conversion	346
Figure 105.	DMA Requests in regular simultaneous mode when MDMA=0b00	347
Figure 106.	DMA requests in regular simultaneous mode when MDMA=0b10	348
Figure 107.	DMA requests in interleaved mode when MDMA=0b10	348
Figure 108.	Temperature sensor channel block diagram	350
Figure 109.	VBAT channel block diagram	352
Figure 110.	VREFINT channel block diagram	352
Figure 111.	DAC1 block diagram	393
Figure 112.	Data registers in single DAC channel mode	394
Figure 113.	Timing diagram for conversion with trigger disabled TEN = 0	395
Figure 114.	DAC LFSR register calculation algorithm	397
Figure 115.	DAC conversion (SW trigger enabled) with LFSR wave generation.	398
Figure 116.	DAC triangle wave generation	398
Figure 117.	DAC conversion (SW trigger enabled) with triangle wave generation	399
Figure 118.	Comparator 1 and 2 block diagrams	407
Figure 119.	Comparator output blanking	411
Figure 120.	STM32F302xB/C/D/E comparator and operational amplifier connections	423
Figure 121.	STM32F302x6/8 comparator and operational amplifier connections	424
Figure 122.	Timer controlled Multiplexer mode	426
Figure 123.	Standalone mode: external gain setting mode	427
Figure 124.	Follower configuration.	428
Figure 125.	PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used	429
Figure 126.	PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering	429
Figure 127.	TSC block diagram	437
Figure 128.	Surface charge transfer analog I/O group structure	438
Figure 129.	Sampling capacitor voltage variation	439
Figure 130.	Charge transfer acquisition sequence	440
Figure 131.	Spread spectrum variation principle	441
Figure 132.	Advanced-control timer block diagram	454
Figure 133.	Counter timing diagram with prescaler division change from 1 to 2	456
Figure 134.	Counter timing diagram with prescaler division change from 1 to 4	456
Figure 135.	Counter timing diagram, internal clock divided by 1	458
Figure 136.	Counter timing diagram, internal clock divided by 2	458
Figure 137.	Counter timing diagram, internal clock divided by 4	459
Figure 138.	Counter timing diagram, internal clock divided by N.	459
Figure 139.	Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)	460
Figure 140.	Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)	460
Figure 141.	Counter timing diagram, internal clock divided by 1	462
Figure 142.	Counter timing diagram, internal clock divided by 2	462
Figure 143.	Counter timing diagram, internal clock divided by 4	463
Figure 144.	Counter timing diagram, internal clock divided by N.	463

Figure 145. Counter timing diagram, update event when repetition counter is not used	464
Figure 146. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6	465
Figure 147. Counter timing diagram, internal clock divided by 2	466
Figure 148. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36	466
Figure 149. Counter timing diagram, internal clock divided by N	467
Figure 150. Counter timing diagram, update event with ARPE=1 (counter underflow)	467
Figure 151. Counter timing diagram, Update event with ARPE=1 (counter overflow)	468
Figure 152. Update rate examples depending on mode and TIMx_RCR register settings	469
Figure 153. Control circuit in normal mode, internal clock divided by 1	470
Figure 154. TI2 external clock connection example	471
Figure 155. Control circuit in external clock mode 1	472
Figure 156. External trigger input block	472
Figure 157. Control circuit in external clock mode 2	473
Figure 158. Capture/compare channel (example: channel 1 input stage)	474
Figure 159. Capture/compare channel 1 main circuit	475
Figure 160. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)	475
Figure 161. Output stage of capture/compare channel (channel 4)	476
Figure 162. Output stage of capture/compare channel (channel 5, idem ch. 6)	476
Figure 163. PWM input mode timing	479
Figure 164. Output compare mode, toggle on OC1	480
Figure 165. Edge-aligned PWM waveforms (ARR=8)	482
Figure 166. Center-aligned PWM waveforms (ARR=8)	483
Figure 167. Generation of 2 phase-shifted PWM signals with 50% duty cycle	484
Figure 168. Combined PWM mode on channel 1 and 3	485
Figure 169. 3-phase combined PWM signals with multiple trigger pulses per period	486
Figure 170. Complementary output with dead-time insertion	487
Figure 171. Dead-time waveforms with delay greater than the negative pulse	488
Figure 172. Dead-time waveforms with delay greater than the positive pulse	488
Figure 173. Various output behavior in response to a break event on BKIN (OSS1 = 1)	492
Figure 174. PWM output state following BKIN and BKIN2 pins assertion (OSS1=1)	493
Figure 175. PWM output state following BKIN assertion (OSS1=0)	494
Figure 176. Clearing TIMx_OCxREF	495
Figure 177. 6-step generation, COM example (OSSR=1)	496
Figure 178. Example of one pulse mode	497
Figure 179. Retriggerable one pulse mode	499
Figure 180. Example of counter operation in encoder interface mode	500
Figure 181. Example of encoder interface mode with TI1FP1 polarity inverted	501
Figure 182. Measuring time interval between edges on 3 signals	502
Figure 183. Example of Hall sensor interface	504
Figure 184. Control circuit in reset mode	505
Figure 185. Control circuit in Gated mode	506
Figure 186. Control circuit in trigger mode	507
Figure 187. Control circuit in external clock mode 2 + trigger mode	508
Figure 188. General-purpose timer block diagram	547
Figure 189. Counter timing diagram with prescaler division change from 1 to 2	549
Figure 190. Counter timing diagram with prescaler division change from 1 to 4	549
Figure 191. Counter timing diagram, internal clock divided by 1	550
Figure 192. Counter timing diagram, internal clock divided by 2	551
Figure 193. Counter timing diagram, internal clock divided by 4	551
Figure 194. Counter timing diagram, internal clock divided by N	552
Figure 195. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)	552
Figure 196. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)	553

Figure 197. Counter timing diagram, internal clock divided by 1	554
Figure 198. Counter timing diagram, internal clock divided by 2	554
Figure 199. Counter timing diagram, internal clock divided by 4	555
Figure 200. Counter timing diagram, internal clock divided by N	555
Figure 201. Counter timing diagram, Update event when repetition counter is not used	556
Figure 202. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6	557
Figure 203. Counter timing diagram, internal clock divided by 2	558
Figure 204. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36	558
Figure 205. Counter timing diagram, internal clock divided by N	559
Figure 206. Counter timing diagram, Update event with ARPE=1 (counter underflow)	559
Figure 207. Counter timing diagram, Update event with ARPE=1 (counter overflow)	560
Figure 208. Control circuit in normal mode, internal clock divided by 1	561
Figure 209. TI2 external clock connection example	561
Figure 210. Control circuit in external clock mode 1	562
Figure 211. External trigger input block	563
Figure 212. Control circuit in external clock mode 2	564
Figure 213. Capture/compare channel (example: channel 1 input stage)	565
Figure 214. Capture/compare channel 1 main circuit	565
Figure 215. Output stage of capture/compare channel (channel 1)	566
Figure 216. PWM input mode timing	568
Figure 217. Output compare mode, toggle on OC1	570
Figure 218. Edge-aligned PWM waveforms (ARR=8)	571
Figure 219. Center-aligned PWM waveforms (ARR=8)	573
Figure 220. Generation of 2 phase-shifted PWM signals with 50% duty cycle	574
Figure 221. Combined PWM mode on channels 1 and 3	575
Figure 222. Clearing TIMx_OCxREF	576
Figure 223. Example of one-pulse mode	577
Figure 224. Retriggerable one pulse mode	579
Figure 225. Example of counter operation in encoder interface mode	580
Figure 226. Example of encoder interface mode with TI1FP1 polarity inverted	581
Figure 227. Control circuit in reset mode	582
Figure 228. Control circuit in gated mode	583
Figure 229. Control circuit in trigger mode	584
Figure 230. Control circuit in external clock mode 2 + trigger mode	585
Figure 231. Master/Slave timer example	586
Figure 232. Gating TIM2 with OC1REF of TIM3	587
Figure 233. Gating TIM2 with Enable of TIM3	588
Figure 234. Triggering TIM2 with update of TIM3	588
Figure 235. Triggering TIM2 with Enable of TIM3	589
Figure 236. Triggering TIM3 and TIM2 with TIM3 TI1 input	590
Figure 237. TIM15 block diagram	618
Figure 238. TIM16 and TIM17 block diagram	619
Figure 239. Counter timing diagram with prescaler division change from 1 to 2	621
Figure 240. Counter timing diagram with prescaler division change from 1 to 4	621
Figure 241. Counter timing diagram, internal clock divided by 1	623
Figure 242. Counter timing diagram, internal clock divided by 2	623
Figure 243. Counter timing diagram, internal clock divided by 4	624
Figure 244. Counter timing diagram, internal clock divided by N	624
Figure 245. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)	625
Figure 246. Counter timing diagram, update event when ARPE=1 (TIMx_ARR	

preloaded).	625
Figure 247. Update rate examples depending on mode and TIMx_RCR register settings	627
Figure 248. Control circuit in normal mode, internal clock divided by 1	628
Figure 249. TI2 external clock connection example	628
Figure 250. Control circuit in external clock mode 1	629
Figure 251. Capture/compare channel (example: channel 1 input stage)	630
Figure 252. Capture/compare channel 1 main circuit	630
Figure 253. Output stage of capture/compare channel (channel 1)	631
Figure 254. Output stage of capture/compare channel (channel 2 for TIM15)	631
Figure 255. PWM input mode timing	633
Figure 256. Output compare mode, toggle on OC1	635
Figure 257. Edge-aligned PWM waveforms (ARR=8)	636
Figure 258. Combined PWM mode on channel 1 and 2	637
Figure 259. Complementary output with dead-time insertion	638
Figure 260. Dead-time waveforms with delay greater than the negative pulse	639
Figure 261. Dead-time waveforms with delay greater than the positive pulse	639
Figure 262. Output behavior in response to a break	642
Figure 263. Example of one pulse mode	643
Figure 264. Measuring time interval between edges on 2 signals	645
Figure 265. Control circuit in reset mode	646
Figure 266. Control circuit in gated mode	647
Figure 267. Control circuit in trigger mode	648
Figure 268. Basic timer block diagram	694
Figure 269. Counter timing diagram with prescaler division change from 1 to 2	696
Figure 270. Counter timing diagram with prescaler division change from 1 to 4	696
Figure 271. Counter timing diagram, internal clock divided by 1	697
Figure 272. Counter timing diagram, internal clock divided by 2	698
Figure 273. Counter timing diagram, internal clock divided by 4	698
Figure 274. Counter timing diagram, internal clock divided by N	699
Figure 275. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)	699
Figure 276. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)	700
Figure 277. Control circuit in normal mode, internal clock divided by 1	701
Figure 278. IR internal hardware connections with TIM16 and TIM17	707
Figure 279. Watchdog block diagram	709
Figure 280. Window watchdog timing diagram	710
Figure 281. Independent watchdog block diagram	714
Figure 282. RTC block diagram	724
Figure 283. I2C block diagram	769
Figure 284. I2C bus protocol	771
Figure 285. Setup and hold timings	773
Figure 286. I2C initialization flowchart	776
Figure 287. Data reception	777
Figure 288. Data transmission	778
Figure 289. Slave initialization flowchart	781
Figure 290. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=0	783
Figure 291. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=1	784
Figure 292. Transfer bus diagrams for I2C slave transmitter	785
Figure 293. Transfer sequence flowchart for slave receiver with NOSTRETCH=0	786
Figure 294. Transfer sequence flowchart for slave receiver with NOSTRETCH=1	787
Figure 295. Transfer bus diagrams for I2C slave receiver	787

Figure 296. Master clock generation	789
Figure 297. Master initialization flowchart	791
Figure 298. 10-bit address read access with HEAD10R=0	791
Figure 299. 10-bit address read access with HEAD10R=1	792
Figure 300. Transfer sequence flowchart for I2C master transmitter for $N \leq 255$ bytes	793
Figure 301. Transfer sequence flowchart for I2C master transmitter for $N > 255$ bytes	794
Figure 302. Transfer bus diagrams for I2C master transmitter	795
Figure 303. Transfer sequence flowchart for I2C master receiver for $N \leq 255$ bytes.	797
Figure 304. Transfer sequence flowchart for I2C master receiver for $N > 255$ bytes	798
Figure 305. Transfer bus diagrams for I2C master receiver	799
Figure 306. Timeout intervals for $t_{\text{LOW:SEXT}}$, $t_{\text{LOW:MEXT}}$	804
Figure 307. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC.	808
Figure 308. Transfer bus diagrams for SMBus slave transmitter (SBC=1)	808
Figure 309. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC	810
Figure 310. Bus transfer diagrams for SMBus slave receiver (SBC=1)	811
Figure 311. Bus transfer diagrams for SMBus master transmitter.	812
Figure 312. Bus transfer diagrams for SMBus master receiver	813
Figure 313. I2C interrupt mapping diagram	818
Figure 314. USART block diagram	840
Figure 315. Word length programming	842
Figure 316. Configurable stop bits	843
Figure 317. TC/TXE behavior when transmitting	845
Figure 318. Start bit detection when oversampling by 16 or 8.	846
Figure 319. Data sampling when oversampling by 16.	850
Figure 320. Data sampling when oversampling by 8.	850
Figure 321. Mute mode using Idle line detection	857
Figure 322. Mute mode using address mark detection	858
Figure 323. Break detection in LIN mode (11-bit break length - LBDL bit is set).	861
Figure 324. Break detection in LIN mode vs. Framing error detection.	862
Figure 325. USART example of synchronous transmission.	863
Figure 326. USART data clock timing diagram (M bits = 00).	863
Figure 327. USART data clock timing diagram (M bits = 01)	864
Figure 328. RX data setup/hold time	864
Figure 329. ISO 7816-3 asynchronous protocol	866
Figure 330. Parity error detection using the 1.5 stop bits.	867
Figure 331. IrDA SIR ENDEC- block diagram	871
Figure 332. IrDA data modulation (3/16) -Normal Mode	871
Figure 333. Transmission using DMA	873
Figure 334. Reception using DMA.	874
Figure 335. Hardware flow control between 2 USARTs.	874
Figure 336. RS232 RTS flow control	875
Figure 337. RS232 CTS flow control	876
Figure 338. USART interrupt mapping diagram	878
Figure 339. SPI block diagram.	904
Figure 340. Full-duplex single master/ single slave application.	905
Figure 341. Half-duplex single master/ single slave application	906
Figure 342. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)	907
Figure 343. Master and three independent slaves.	908
Figure 344. Hardware/software slave select management	909
Figure 345. Data clock timing diagram.	911
Figure 346. Data alignment when data length is not equal to 8-bit or 16-bit	912

Figure 347. Packing data in FIFO for transmission and reception	916
Figure 348. Master full duplex communication.	919
Figure 349. Slave full duplex communication.	920
Figure 350. Master full duplex communication with CRC	921
Figure 351. Master full duplex communication in packed mode	922
Figure 352. NSSP pulse generation in Motorola SPI master mode	925
Figure 353. TI mode transfer	926
Figure 354. I ² S block diagram	929
Figure 355. I2S full duplex block diagram	931
Figure 356. I ² S Philips protocol waveforms (16/32-bit full accuracy, CPOL = 0).	932
Figure 357. I ² S Philips standard waveforms (24-bit frame with CPOL = 0).	932
Figure 358. Transmitting 0x8EAA33	933
Figure 359. Receiving 0x8EAA33	933
Figure 360. I ² S Philips standard (16-bit extended to 32-bit packet frame with CPOL = 0)	933
Figure 361. Example of 16-bit data frame extended to 32-bit channel frame	933
Figure 362. MSB Justified 16-bit or 32-bit full-accuracy length with CPOL = 0	934
Figure 363. MSB justified 24-bit frame length with CPOL = 0	934
Figure 364. MSB justified 16-bit extended to 32-bit packet frame with CPOL = 0	935
Figure 365. LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0	935
Figure 366. LSB justified 24-bit frame length with CPOL = 0.	935
Figure 367. Operations required to transmit 0x3478AE.	936
Figure 368. Operations required to receive 0x3478AE	936
Figure 369. LSB justified 16-bit extended to 32-bit packet frame with CPOL = 0	936
Figure 370. Example of 16-bit data frame extended to 32-bit channel frame	937
Figure 371. PCM standard waveforms (16-bit)	937
Figure 372. PCM standard waveforms (16-bit extended to 32-bit packet frame).	938
Figure 373. Audio sampling frequency definition	938
Figure 374. I ² S clock generator architecture	939
Figure 375. CAN network topology	961
Figure 376. bxCAN operating modes.	964
Figure 377. bxCAN in silent mode	965
Figure 378. bxCAN in loop back mode	965
Figure 379. bxCAN in combined mode	966
Figure 380. Transmit mailbox states	967
Figure 381. Receive FIFO states	968
Figure 382. Filter bank scale configuration - register organization	971
Figure 383. Example of filter numbering	972
Figure 384. Filtering mechanism - example.	973
Figure 385. CAN error state diagram.	974
Figure 386. Bit timing.	976
Figure 387. CAN frames	977
Figure 388. Event flags and interrupt generation.	978
Figure 389. Can mailbox registers	990
Figure 390. USB peripheral block diagram	1007
Figure 391. Packet buffer areas with examples of buffer description table locations	1011
Figure 392. Block diagram of STM32 MCU and Cortex-M4 [®] F-level debug support	1039
Figure 393. SWJ debug port	1041
Figure 394. JTAG TAP connections.	1045
Figure 395. TPIU block diagram	1061

1 Overview of the manual

Table 1. Available features related to each product

Peripherals	STM32F302xB/C	STM32F302xD/E	STM32F302x6/8
Section 10: Reset and clock control (RCC)	Available	Available	Available
Section 11: General-purpose I/Os (GPIO)	Up to 87	Up to 115	Up to 52
Section 13: Direct memory access controller (DMA)	DMA1&2	DMA1&2	DMA1
Section 15: Analog-to-digital converters (ADC)	ADC1&2	ADC1&2	ADC1
Section 16: Digital-to-analog converter (DAC1)	DAC1 Ch.1	DAC1 Ch.1	DAC1 Ch.1
Section 17: Comparator (COMP)	Comp 1,2,4&6	Comp 1,2,4&6	Comp 2,4&6
Section 18: Operational amplifier (OPAMP)	Opamp1&2	Opamp1&2	Opamp2
Section 19: Touch sensing controller (TSC)	Up to 24	Up to 24	Up to 17
Section 20: Advanced-control timers (TIM1)	TIM1	TIM1	TIM1
Section 21: General-purpose timers (TIM2/TIM3/TIM4)	TIM2,3&4	TIM2,3&4	TIM2
Section 22: General-purpose timers (TIM15/16/17)	TIM15,16&17	TIM15,16&17	TIM15,16&17
Section 23: Basic timers (TIM6)	TIM6	TIM6	TIM6
Section 24: Infrared interface (IRTIM)	Available	Available	Available
Section 26: Independent watchdog (IWDG)	Available	Available	Available
Section 25: System window watchdog (WWDG)	Available	Available	Available
Section 27: Real-time clock (RTC)	Available	Available	Available
Section 28: Inter-integrated circuit (I2C) interface	I2C1 and I2C2	I2C1, I2C2 and I2C3	I2C1, I2C2 and I2C3
Section 29: Universal synchronous asynchronous receiver transmitter (USART)	3 USARTs and up to 2 UARTS	3 USARTs and up to 2 UARTS	Up to 3 USARTs

Table 1. Available features related to each product (continued)

Peripherals	STM32F302xB/C	STM32F302xD/E	STM32F302x6/8
<i>Section 30: Serial peripheral interface / inter-IC sound (SPI/I2S)</i>	SPI1, SPI2, SPI3	SPI1, SPI2, SPI3, SPI4	SPI2, SPI3 with I2S
<i>Section 31: Controller area network (bxCAN)</i>	Available	Available	Available
<i>Section 32: Universal serial bus full-speed device interface (USB)</i>	Available	Available	Available

2 Documentation conventions

2.1 List of abbreviations for registers

The following abbreviations are used in register descriptions:

read/write (rw)	Software can read and write to these bits.
read-only (r)	Software can only read these bits.
write-only (w)	Software can only write to this bit. Reading the bit returns the reset value.
read/clear (rc_w1)	Software can read as well as clear this bit by writing 1. Writing '0' has no effect on the bit value.
read/clear (rc_w0)	Software can read as well as clear this bit by writing 0. Writing '1' has no effect on the bit value.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to '0'. Writing '0' has no effect on the bit value.
read/set (rs)	Software can read as well as set this bit. Writing '0' has no effect on the bit value.

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

2.2 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.
- **IAP (in-application programming):** IAP is the ability to re-program the Flash memory of a microcontroller while the user program is running.
- **ICP (in-circuit programming):** ICP is the ability to program the Flash memory of a microcontroller using the JTAG protocol, the SWD protocol or the bootloader while the device is mounted on the user application board.
- **Option bytes:** product configuration bits stored in the Flash memory.
- **OBL:** option byte loader.
- **AHB:** advanced high-performance bus.

2.3 Peripheral availability

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

3 System and memory overview

3.1 System architecture

The STM32F302xB/C/D/E and STM32F302x6/8 main system consists of:

- Five masters:
 - Cortex[®]-M4 core I-bus
 - Cortex[®]-M4 core D-bus
 - Cortex[®]-M4 core S-bus
 - GP-DMA1 and GP-DMA2 (general-purpose DMA)
- Six (seven in STM32F302xD/E) slaves:
 - Internal Flash memory on the DCode
 - Internal Flash memory on ICode
 - Up to Internal 40 Kbyte SRAM
 - FMC in STM32F302xD/E
 - AHB to APBx (APB1 or APB2), which connect all the APB peripherals
 - AHB dedicated to GPIO ports
 - ADCs 1, 2.

The STM32F302x6/8 main system consists of:

- Four masters:
 - Cortex[®]-M4 core I-bus
 - Cortex[®]-M4 core D-bus
 - Cortex[®]-M4 core S-bus
 - GP-DMA1 (general-purpose DMA)
- Six slaves:
 - Internal Flash memory on the DCode
 - Internal Flash memory on ICode
 - Up to Internal 16 Kbyte SRAM
 - AHB to APBx (APB1 or APB2), which connect all the APB peripherals
 - AHB dedicated to GPIO ports
 - ADC 1

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

Figure 1. STM32F302xB/C system architecture

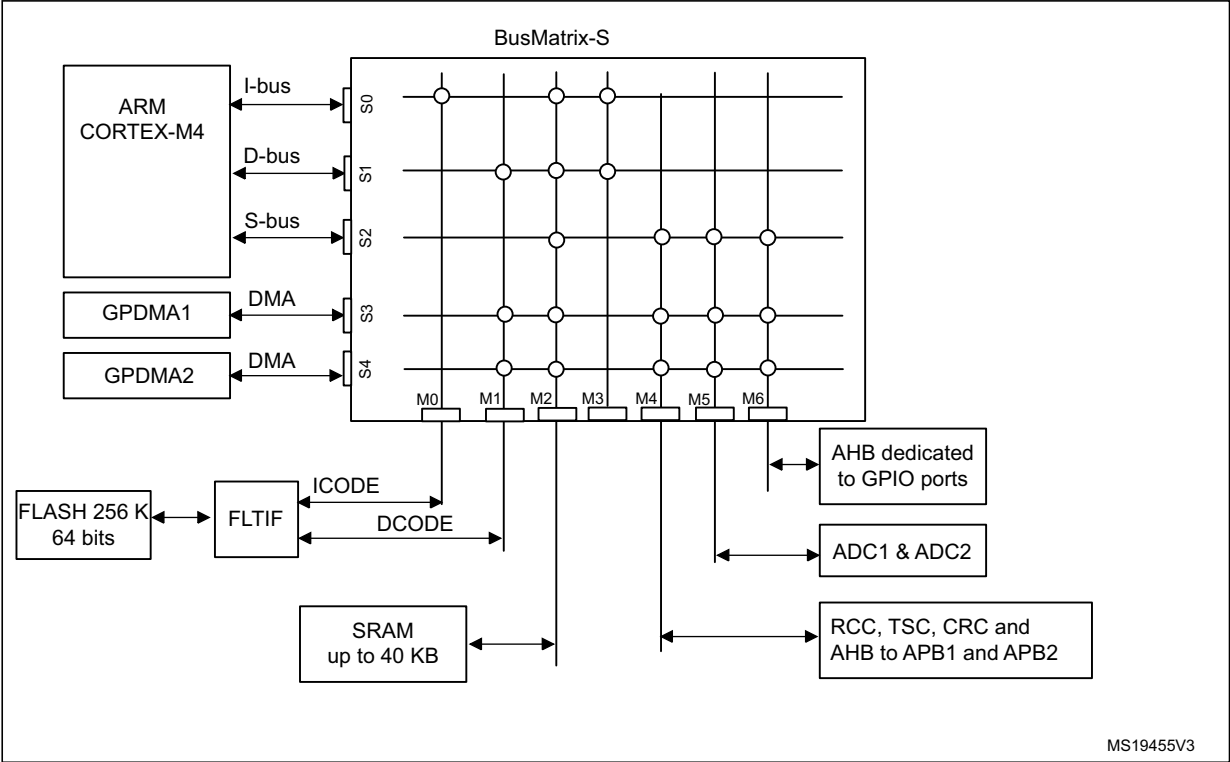


Figure 2. STM32F302x6/8 system architecture

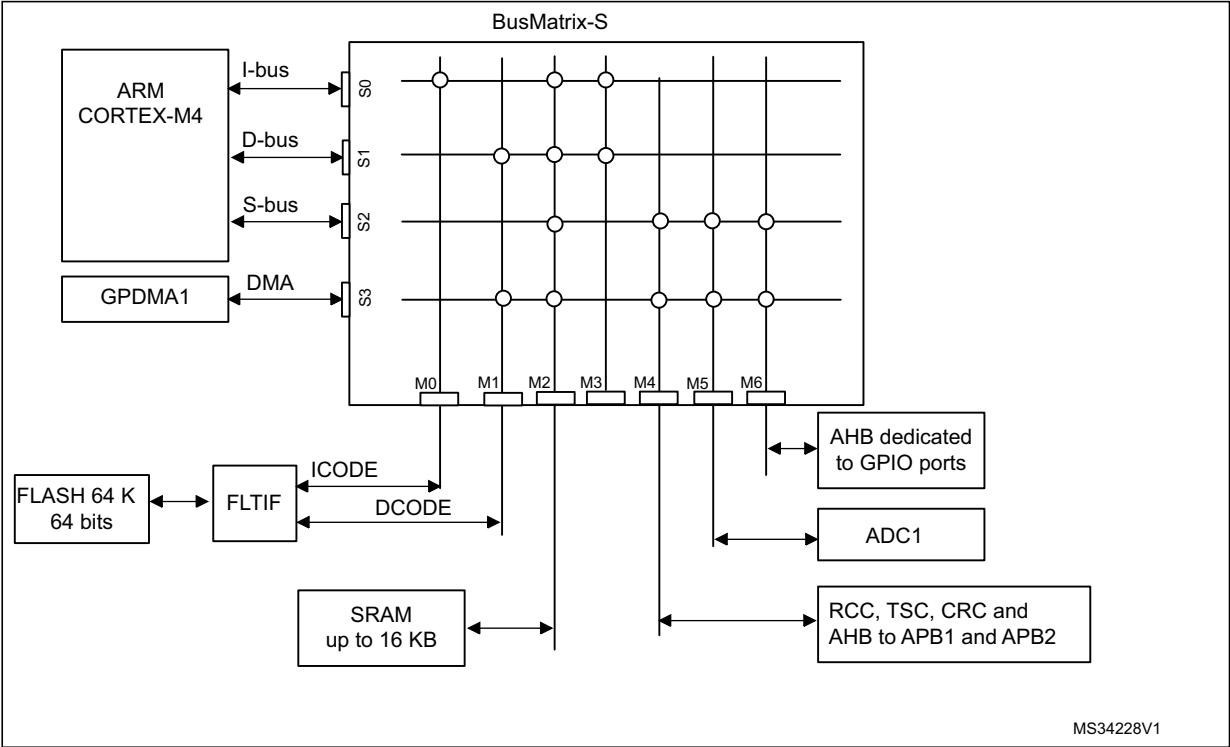
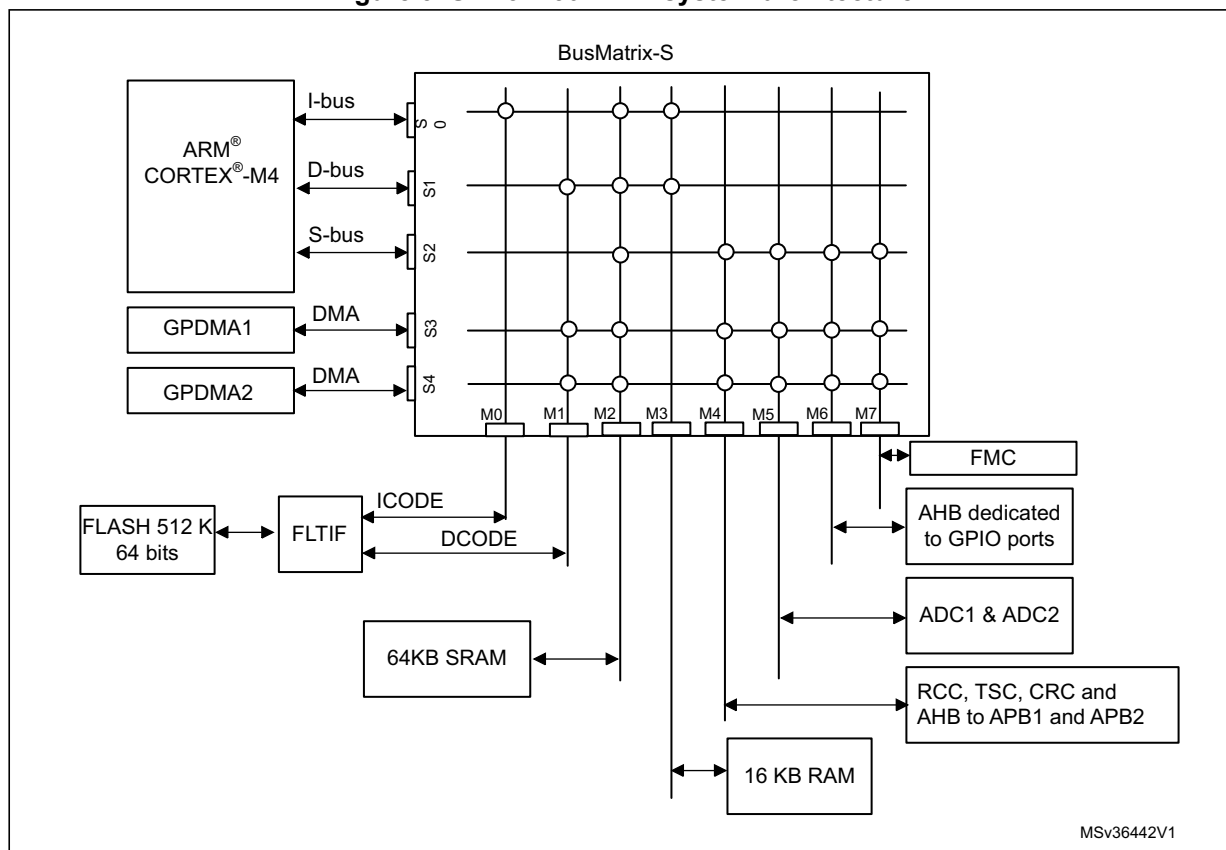


Figure 3. STM32F302xD/E system architecture



MSv36442V1

3.1.1 S0: I-bus

This bus connects the Instruction bus of the Cortex®-M4 core to the BusMatrix. This bus is used by the core to fetch instructions. The targets of this bus are the internal Flash memory and the SRAM.

3.1.2 S1: D-bus

This bus connects the DCode bus (literal load and debug access) of the Cortex®-M4 core to the BusMatrix. The targets of this bus are the internal Flash memory and the SRAM.

3.1.3 S2: S-bus

This bus connects the system bus of the Cortex®-M4 core to the BusMatrix. This bus is used to access data located in the peripheral or SRAM area. The targets of this bus are the SRAM, the AHB to APB1/APB2 bridges, the AHB IO port and the 2 ADCs.

3.1.4 S3, S4: DMA-bus

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

3.1.5 BusMatrix

The BusMatrix manages the access arbitration between Masters. The arbitration uses a Round Robin algorithm. The BusMatrix is composed of five masters (CPU AHB, System bus, DCode bus, ICode bus, DMA1&2 bus) and seven slaves (FLITF, SRAM, AHB2GPIO and AHB2APB1/2 bridges, and ADCs).

AHB/APB bridges

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

Refer to [Section 3.2.2: Memory map and register boundary addresses on page 48](#) for the address mapping of the peripherals connected to this bridge.

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

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.

3.2 Memory organization

3.2.1 Introduction

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

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

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

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

3.2.2 Memory map and register boundary addresses

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

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

Table 2. STM32F302xB/C peripheral register boundary addresses

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB3	0x5000 0000 - 0x5000 03FF	1 K	ADC1 - ADC2	Section 15.6.4 on page 388
	0x4800 1800 - 0x4FFF FFFF	~132 M	Reserved	
AHB2	0x4800 1400 - 0x4800 17FF	1 K	GPIOF	Section 11.4.12 on page 231
	0x4800 1000 - 0x4800 13FF	1 K	GPIOE	
	0x4800 0C00 - 0x4800 0FFF	1 K	GPIOD	
	0x4800 0800 - 0x4800 0BFF	1 K	GPIOC	
	0x4800 0400 - 0x4800 07FF	1 K	GPIOB	
	0x4800 0000 - 0x4800 03FF	1 K	GPIOA	
	0x4002 4400 - 0x47FF FFFF	~128 M	Reserved	

Table 2. STM32F302xB/C peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB1	0x4002 4000 - 0x4002 43FF	1 K	TSC	Section 19.6.11 on page 451
	0x4002 3400 - 0x4002 3FFF	3 K	Reserved	
	0x4002 3000 - 0x4002 33FF	1 K	CRC	Section 6.4.6 on page 87
	0x4002 2400 - 0x4002 2FFF	3 K	Reserved	
	0x4002 2000 - 0x4002 23FF	1 K	Flash interface	Section 4.6 on page 77
	0x4002 1400 - 0x4002 1FFF	3 K	Reserved	
	0x4002 1000 - 0x4002 13FF	1 K	RCC	Section 10.4.14 on page 214
	0x4002 0800 - 0x4002 0FFF	2 K	Reserved	
	0x4002 0400 - 0x4002 07FF	1 K	DMA2	Section 13.5.7 on page 264
	0x4002 0000 - 0x4002 03FF	1 K	DMA1	
	0x4001 8000 - 0x4001 FFFF	32 K	Reserved	
APB2	0x4001 4C00 - 0x4001 7FFF	13 K	Reserved	
	0x4001 4800 - 0x4001 4BFF	1 K	TIM17	Section 22.6.17 on page 692
	0x4001 4400 - 0x4001 47FF	1 K	TIM16	
	0x4001 4000 - 0x4001 43FF	1 K	TIM15	Section 22.5.18 on page 672
	0x4001 3C00 - 0x4001 3FFF	1 K	Reserved	
	0x4001 3800 - 0x4001 3BFF	1 K	USART1	Section 25.7.12 on page 685
	0x4001 3400 - 0x4001 37FF	1 K	Reserved	
	0x4001 3000 - 0x4001 33FF	1 K	SPI1	Section 30.9.10 on page 959
	0x4001 2C00 - 0x4001 2FFF	1 K	TIM1	Section 20.4.25 on page 543
	0x4001 0800 - 0x4001 2BFF	8 K	Reserved	
	0x4001 0400 - 0x4001 07FF	1 K	EXTI	Section 14.3.13 on page 284
	0x4001 0000 - 0x4001 03FF	1 K	SYSCFG + COMP + OPAMP	Section 12.1.7 on page 243, Section 17.5.5 on page 421, Section 18.4.3 on page 435
	0x4000 9C00 - 0x4000 FFFF	25 K	Reserved	

Table 2. STM32F302xB/C peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1	0x4000 7800 - 0x4000 9BFF	9 K	Reserved	
	0x4000 7400 - 0x4000 77FF	1 K	DAC1	Section 16.7.8 on page 405
	0x4000 7000 - 0x4000 73FF	1 K	PWR	Section 8.4.3 on page 162
	0x4000 6C00 - 0x4000 6FFF	1 K	Reserved	
	0x4000 6800 - 0x4000 6BFF	1 K	Reserved	
	0x4000 6400 - 0x4000 67FF	1 K	bxCAN	Section 31.9.5 on page 1001
	0x4000 6000 - 0x4000 63FF	1 K	USB SRAM	Section 32.6.4 on page 1037
	0x4000 5C00 - 0x4000 5FFF	1 K	USB device FS	
	0x4000 5800 - 0x4000 5BFF	1 K	I2C2	Section 28.7.12 on page 834
	0x4000 5400 - 0x4000 57FF	1 K	I2C1	
	0x4000 5000 - 0x4000 53FF	1 K	UART5	Section 25.7.12 on page 685
	0x4000 4C00 - 0x4000 4FFF	1 K	UART4	
	0x4000 4800 - 0x4000 4BFF	1 K	USART3	
	0x4000 4400 - 0x4000 47FF	1 K	USART2	
	0x4000 3400 - 0x4000 43FF	4 K	Reserved	
	0x4000 3000 - 0x4000 33FF	1 K	IWDG	Section 26.4.6 on page 721
	0x4000 2C00 - 0x4000 2FFF	1 K	WWDG	Section 25.4.4 on page 713
	0x4000 2800 - 0x4000 2BFF	1 K	RTC	Section 27.6.20 on page 765
	0x4000 1400 - 0x4000 27FF	5 K	Reserved	
	0x4000 1000 - 0x4000 13FF	1 K	TIM6	Section 23.4.9 on page 706
	0x4000 0C00 - 0x4000 0FFF	1 K	Reserved	
	0x4000 0800 - 0x4000 0BFF	1 K	TIM4	Section 21.4.19 on page 614
	0x4000 0400 - 0x4000 07FF	1 K	TIM3	
	0x4000 0000 - 0x4000 03FF	1 K	TIM2	
	0x2000 A000 - 3FFF FFFF	~512 M	Reserved	
	0x2000 0000 - 0x2000 9FFF	40 K	SRAM	
	0x1FFF F800 - 0x1FFF FFFF	2 K	Option bytes	
	0x1FFF D800 - 0x1FFF F7FF	8 K	System memory	
	0x0804 0000 - 0x1FFF D7FF	~384 M	Reserved	
	0x0800 0000 - 0x0803 FFFF	256 K	Main Flash memory	
	0x0004 0000 - 0x07FF FFFF	~128 M	Reserved	
	0x0000 000 - 0x0003 FFFF	256 K	Main Flash memory, system memory or SRAM depending on BOOT configuration	

Table 3. STM32F302xD/E peripheral register boundary addresses

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB4	0xA000 0000 - 0xA000 0FFF	4 K	FMC control registers	Section 7 on page 88
	0x8000 0000 - 0x9FFF FFFF	512 M	FMC Banks 3 and 4	
	0x6000 0000 - 0x7FFF FFFF	512 M	FMC Banks 1 and 2	
AHB3	0x5000 0000 - 0x5000 03FF	1 K	ADC1 - ADC2	Section 15.6.4 on page 388
	0x4800 2000 - 0x4FFF FFFF	~132 M	Reserved	
AHB2	0x4800 1C00 - 0x4800 1FFF	1 K	GPIOH	Section 11.4.12 on page 231
	0x4800 1800 - 0x4800 1BFF	1 K	GPIOG	
	0x4800 1400 - 0x4800 17FF	1 K	GPIOF	
	0x4800 1000 - 0x4800 13FF	1 K	GPIOE	
	0x4800 0C00 - 0x4800 0FFF	1 K	GPIOD	
	0x4800 0800 - 0x4800 0BFF	1 K	GPIOC	
	0x4800 0400 - 0x4800 07FF	1 K	GPIOB	
	0x4800 0000 - 0x4800 03FF	1 K	GPIOA	
	0x4002 4400 - 0x47FF FFFF	~128 M	Reserved	
AHB1	0x4002 4000 - 0x4002 43FF	1 K	TSC	Section 19.6.11 on page 451
	0x4002 3400 - 0x4002 3FFF	3 K	Reserved	
	0x4002 3000 - 0x4002 33FF	1 K	CRC	Section 6.4.6 on page 87
	0x4002 2400 - 0x4002 2FFF	3 K	Reserved	
	0x4002 2000 - 0x4002 23FF	1 K	Flash interface	Section 4.6 on page 77
	0x4002 1400 - 0x4002 1FFF	3 K	Reserved	
	0x4002 1000 - 0x4002 13FF	1 K	RCC	Section 10.4.14 on page 214
	0x4002 0800 - 0x4002 0FFF	2 K	Reserved	
	0x4002 0400 - 0x4002 07FF	1 K	DMA2	Section 13.5.7 on page 264
	0x4002 0000 - 0x4002 03FF	1 K	DMA1	
	0x4001 8000 - 0x4001 FFFF	32 K	Reserved	

Table 3. STM32F302xD/E peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB2	0x4001 4C00 - 0x4001 7FFF	13 K	Reserved	
	0x4001 4800 - 0x4001 4BFF	1 K	TIM17	Section 22.6.17 on page 692
	0x4001 4400 - 0x4001 47FF	1 K	TIM16	
	0x4001 4000 - 0x4001 43FF	1 K	TIM15	Section 22.5.18 on page 672
	0x4001 3C00 - 0x4001 3FFF	1 K	SPI4	Section 30.9.10 on page 959
	0x4001 3800 - 0x4001 3BFF	1 K	USART1	Section 25.7.12 on page 685
	0x4001 3400 - 0x4001 37FF	1 K	Reserved	
	0x4001 3000 - 0x4001 33FF	1 K	SPI1	Section 30.9.10 on page 959
	0x4001 2C00 - 0x4001 2FFF	1 K	TIM1	Section 20.4.25 on page 543
	0x4001 0800 - 0x4001 2BFF	8 K	Reserved	
	0x4001 0400 - 0x4001 07FF	1 K	EXTI	Section 14.3.13 on page 284
	0x4001 0000 - 0x4001 03FF	1 K	SYSCFG + COMP + OPAMP	Section 12.1.7 on page 243, Section 17.5.5 on page 421, Section 18.4.3 on page 435
	0x4000 9C00 - 0x4000 FFFF	25 K	Reserved	

Table 3. STM32F302xD/E peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1	0x4000 7800 - 0x4000 9BFF	9 K	Reserved	
	0x4000 7400 - 0x4000 77FF	1 K	DAC1	Section 16.7.8 on page 405
	0x4000 7000 - 0x4000 73FF	1 K	PWR	Section 8.4.3 on page 162
	0x4000 6C00 - 0x4000 6FFF	1 K	Reserved	
	0x4000 6800 - 0x4000 6BFF	1 K	Reserved	
	0x4000 6400 - 0x4000 67FF	1 K	bxCAN	Section 31.9.5 on page 1001
	0x4000 6000 - 0x4000 63FF	1 K	USB/CAN SRAM	Section 32.6.4 on page 1037
	0x4000 5C00 - 0x4000 5FFF	1 K	USB device FS	
	0x4000 5800 - 0x4000 5BFF	1 K	I2C2	Section 28.7.12 on page 834
	0x4000 5400 - 0x4000 57FF	1 K	I2C1	
	0x4000 5000 - 0x4000 53FF	1 K	UART5	Section 25.7.12 on page 685
	0x4000 4C00 - 0x4000 4FFF	1 K	UART4	
	0x4000 4800 - 0x4000 4BFF	1 K	USART3	
	0x4000 4400 - 0x4000 47FF	1 K	USART2	
	0x4000 3400 - 0x4000 43FF	4 K	Reserved	
	0x4000 3000 - 0x4000 33FF	1 K	IWDG	Section 26.4.6 on page 721
	0x4000 2C00 - 0x4000 2FFF	1 K	WWDG	Section 25.4.4 on page 713
	0x4000 2800 - 0x4000 2BFF	1 K	RTC	Section 27.6.20 on page 765
	0x4000 1400 - 0x4000 27FF	5 K	Reserved	
	0x4000 1000 - 0x4000 13FF	1 K	TIM6	Section 23.4.9 on page 706
	0x4000 0C00 - 0x4000 0FFF	1 K	Reserved	
	0x4000 0800 - 0x4000 0BFF	1 K	TIM4	Section 21.4.19 on page 614
	0x4000 0400 - 0x4000 07FF	1 K	TIM3	
	0x4000 0000 - 0x4000 03FF	1 K	TIM2	
	0x2000 A000 - 3FFF FFFF	~512 M	Reserved	
	0x2000 0000 - 0x2000 FFFF	64 K	SRAM	
	0x1FFF F800 - 0x1FFF FFFF	2 K	Option bytes	
	0x1FFF D800 - 0x1FFF F7FF	8 K	System memory	
	0x0804 0000 - 0x1FFF D7FF	~384 M	Reserved	
	0x0800 0000 - 0x0807 FFFF	512 K	Main Flash memory	
	0x0004 0000 - 0x07FF FFFF	~128 M	Reserved	
	0x0000 000 - 0x0007 FFFF	512 K	Main Flash memory, system memory or SRAM depending on BOOT configuration	

Table 4. STM32F302x6/x8 peripheral register boundary addresses

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB3	0x5000 0000 - 0x5000 03FF	1 K	ADC1	Section 15.6.4 on page 388
	0x4800 1800 - 0x4FFF FFFF	~132 M	Reserved	
AHB2	0x4800 1400 - 0x4800 17FF	1 K	GPIOF	Section 11.4.12 on page 231
	0x4800 1000 - 0x4800 13FF	1 K	Reserved	
	0x4800 0C00 - 0x4800 0FFF	1 K	GPIOD	Section 11.4.12 on page 231
	0x4800 0800 - 0x4800 0BFF	1 K	GPIOC	
	0x4800 0400 - 0x4800 07FF	1 K	GPIOB	
	0x4800 0000 - 0x4800 03FF	1 K	GPIOA	
	0x4002 4400 - 0x47FF FFFF	~128 M	Reserved	
AHB1	0x4002 4000 - 0x4002 43FF	1 K	TSC	Section 19.6.11 on page 451
	0x4002 3400 - 0x4002 3FFF	3 K	Reserved	
	0x4002 3000 - 0x4002 33FF	1 K	CRC	Section 6.4.6 on page 87
	0x4002 2400 - 0x4002 2FFF	3 K	Reserved	
	0x4002 2000 - 0x4002 23FF	1 K	Flash interface	Section 4.6 on page 77
	0x4002 1400 - 0x4002 1FFF	3 K	Reserved	
	0x4002 1000 - 0x4002 13FF	1 K	RCC	Section 10.4.14 on page 214
	0x4002 0400 - 0x4002 0FFF	3 K	Reserved	
	0x4002 0000 - 0x4002 03FF	1 K	DMA1	Section 13.5.7 on page 264
	0x4001 8000 - 0x4001 FFFF	32 K	Reserved	
APB2	0x4001 4C00 - 0x4001 7FFF	13 K	Reserved	
	0x4001 4800 - 0x4001 4BFF	1 K	TIM17	Section 22.6.17 on page 692
	0x4001 4400 - 0x4001 47FF	1 K	TIM16	
	0x4001 4000 - 0x4001 43FF	1 K	TIM15	Section 22.5.18 on page 672
	0x4001 3C00 - 0x4001 3FFF	1 K	Reserved	
	0x4001 3800 - 0x4001 3BFF	1 K	USART1	Section 25.7.12 on page 685
	0x4001 2C00 - 0x4001 2FFF	1 K	TIM1	Section 20.4.25 on page 543
	0x4001 0800 - 0x4001 2BFF	8 K	Reserved	
	0x4001 0400 - 0x4001 07FF	1 K	EXTI	Section 14.3.13 on page 284
	0x4001 0000 - 0x4001 03FF	1 K	SYSCFG + COMP + OPAMP	Section 12.1.7 on page 243 , Section 17.5.5 on page 421 , Section 18.4.3 on page 435
	0x4000 9C00 - 0x4000 FFFF	25 K	Reserved	

Table 4. STM32F302x6/x8 peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1	0x4000 7C00 - 0x4000 9BFF	8 K	Reserved	
	0x4000 7800 - 0x4000 7BFF	1 K	I2C3	Section 28.7.12 on page 834
	0x4000 7400 - 0x4000 77FF	1 K	DAC1	Section 16.7.8 on page 405
	0x4000 7000 - 0x4000 73FF	1 K	PWR	Section 8.4.3 on page 162
	0x4000 6800 - 0x4000 6FFF	2 K	Reserved	
	0x4000 6400 - 0x4000 67FF	1 K	bxCAN	Section 31.9.5 on page 1001
	0x4000 6000 - 0x4000 63FF	1 K	USB/CAN SRAM	Section 32.6.4 on page 1037
	0x4000 5C00 - 0x4000 5FFF	1 K	USB device FS	
	0x4000 5800 - 0x4000 5BFF	1 K	I2C2	Section 28.7.12 on page 834
	0x4000 5400 - 0x4000 57FF	1 K	I2C1	
	0x4000 4C00 - 0x4000 53FF	2 K	Reserved	
	0x4000 4800 - 0x4000 4BFF	1 K	USART3	Section 25.7.12 on page 685
	0x4000 4400 - 0x4000 47FF	1 K	USART2	
	0x4000 4000 - 0x4000 43FF	1 K	I2S3ext	Section 30.9.10 on page 959
	0x4000 3C00 - 0x4000 3FFF	1 K	SPI3/I2S3	
	0x4000 3800 - 0x4000 3BFF	1 K	SPI2/I2S2	
	0x4000 3400 - 0x4000 37FF	1 K	I2S2ext	
	0x4000 3000 - 0x4000 33FF	1 K	IWDG	Section 26.4.6 on page 721
	0x4000 2C00 - 0x4000 2FFF	1 K	WWDG	Section 25.4.4 on page 713
	0x4000 2800 - 0x4000 2BFF	1 K	RTC	Section 27.6.20 on page 765
	0x4000 1400 - 0x4000 27FF	5 K	Reserved	
	0x4000 1000 - 0x4000 13FF	1 K	TIM6	Section 23.4.9 on page 706
	0x4000 0400 - 0x4000 0FFF	3 K	Reserved	
	0x4000 0000 - 0x4000 03FF	1 K	TIM2	Section 21.4.19 on page 614
	0x2000 4000 - 3FFF FFFF	~512 M	Reserved	
	0x2000 0000 - 0x2000 3FFF	16 K	SRAM	
	0x1FFF F800 - 0x1FFF FFFF	2 K	Option bytes	
	0x1FFF D800 - 0x1FFF F7FF	8 K	System memory	
	0x0801 0000 - 0x1FFF D7FF	~384 M	Reserved	
	0x0800 0000 - 0x0800 FFFF	64 K	Main Flash memory	
	0x0001 0000 - 0x07FF FFFF	~128 M	Reserved	
	0x0000 000 - 0x0000 FFFF	64 K	Main Flash memory, system memory or SRAM depending on BOOT configuration	

3.3 Embedded SRAM

STM32F302xB/C devices feature up to 40 Kbytes of static SRAM. It can be accessed as bytes, halfwords (16 bits) or full words (32 bits). It can be addressed at maximum system clock frequency without wait states and can be accessed by both CPU and DMA.

STM32F302x6/8 devices feature only up to 16 Kbytes of static SRAM.

STM32F302xD/E devices feature up to 64 Kbytes of static SRAM.

3.3.1 Parity check

On the STM32F302xB/C devices, for the 40-Kbyte SRAM, a parity check is implemented only on the first 16 Kbytes. On the STM32F302xD/E devices, the parity check is implemented on the first 32 Kbytes of SRAM. On the STM32F302x6/x8 the SRAM parity check is not supported. The SRAM parity check is disabled by default. It is enabled by the user, when needed, using an option bit.

The data bus width of the SRAM supporting the parity check 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 on data and address and stored when writing into the SRAM. Then, they are automatically checked when reading. If one bit fails, an NMI is generated if the SRAM parity check is enabled. The same error can also be linked to the Break input of TIMER 1, 15, 16 and 17, by setting the SRAM_PARITY_LOCK control bit in the SYSCFG configuration register 2 (SYSCFG_CFGR2). In case of parity error, the SRAM Parity Error flag (SRAM_PEF) is set in the SYSCFG configuration register 2 (SYSCFG_CFGR2). For more details, please refer to the SYSCFG configuration register 2 (SYSCFG_CFGR2).

The BYP_ADD_PAR bit in SYSCFG_CFGR2 register can be used to prevent an unwanted parity error to occur when the user programs a code in the RAM at address 0x2XXXXXXX (address in the address range 0x20000000-0x20002000) and then executes the code from RAM at boot (RAM is remapped at address 0x00).

3.4 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 two parts:
 - Option bytes for hardware and memory protection user configuration.
 - System memory which contains the proprietary boot loader code. Please, refer to [Section 4: Embedded Flash memory](#) for more details.

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

3.5 Boot configuration

In the STM32F302xx, three different boot modes can be selected through the BOOT0 pin and nBOOT1 bit in the User option byte, as shown in the following table:

Table 5. Boot modes

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

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

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

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x0800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x0800 0000.
- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x1FFF D800).
- Boot from the embedded SRAM: the SRAM is aliased in the boot memory space (0x0000 0000), but it is still accessible from its original memory space (0x2000 0000).

3.5.1 Embedded boot loader

The embedded boot loader is located in the System memory, programmed by ST during production. It is used to reprogram the Flash memory through

- USART1(PA9/PA10), USART2(PD5/PD6) or USB(PA11/PA12) on STM32F302xB/C devices,
- USART1(PA9/PA10), USART2(PA2/PA3) or USB(PA11/PA12) on STM32F302x6/8 and STM32F302xD/E devices.

4 Embedded Flash memory

4.1 Flash main features

Up to 512 Kbytes of Flash memory in STM32F302xD/E, up to 256 Kbytes of Flash memory in STM32F302xB/C devices and up to 64 Kbytes of Flash memory in STM32F302x6/8 devices.

- Memory organization:
 - Main memory block:
64 Kbit x 64 bits in STM32F302xD/E, 32_Kbits × 64 bits in STM32F302xB/C devices.
8 Kbit x 64 bits in STM32F302x6/8 devices.
 - Information block:
1280 × 64 bits

Flash memory interface (FLITF) features:

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

4.2 Flash memory functional description

4.2.1 Flash memory organization

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

The memory organization is based on a main memory block containing 128 pages of 2 Kbytes in STM32F302xB/C devices, 256 pages of 2 Kbytes in STM32F302xD/E devices, 32 pages of 2 Kbytes in the STM32F302x6/8. and an information block as shown in [Table 6](#).

Table 6. Flash module organization⁽¹⁾

Flash area	Flash memory addresses	Size (bytes)	Name
Main memory	0x0800 0000 - 0x0800 07FF	2 K	Page 0
	0x0800 0800 - 0x0800 0FFF	2 K	Page 1
	0x0800 1000 - 0x0800 17FF	2 K	Page 2
	0x0800 1800 - 0x0800 1FFF	2 K	Page 3
	.	.	.
	.	.	.
	.	.	.
Information block	0x0803 F800 - 0x0807 FFFF	2 K	Page 255
	0x1FFF D800 - 0x1FFF F7FF	8 K	System memory
	0x1FFF F800 - 0x1FFF F80F	16	Option bytes
Flash memory interface registers	0x4002 2000 - 0x4002 2003	4	FLASH_ACR
	0x4002 2004 - 0x4002 2007	4	FLASH_KEYR
	0x4002 2008 - 0x4002 200B	4	FLASH_OPTKEYR
	0x4002 200C - 0x4002 200F	4	FLASH_SR
	0x4002 2010 - 0x4002 2013	4	FLASH_CR
	0x4002 2014 - 0x4002 2017	4	FLASH_AR
	0x4002 2018 - 0x4002 201B	4	Reserved
	0x4002 201C - 0x4002 201F	4	FLASH_OBR
	0x4002 2020 - 0x4002 2023	4	FLASH_WRP

1. The gray color is used for reserved Flash memory addresses.

The information block is divided into two parts:

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

4.2.2 Read operations

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

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

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

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

Instruction fetch

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

Prefetch buffer

The prefetch buffer is 2 blocks wide where each block consists of 8 bytes. The prefetch blocks are direct-mapped. A block can be completely replaced on a single read to the Flash memory as the size of the block matches the bandwidth of the Flash memory.

The implementation of this prefetch buffer makes a faster CPU execution possible as the CPU fetches one word at a time with the next word readily available in the prefetch buffer. This implies that the acceleration ratio is in the order of 2 assuming that the code is aligned at a 64-bit boundary for the jumps.

Prefetch controller

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

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

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

If there is not any high frequency clock available in the system, Flash memory accesses can be made on a half cycle of HCLK (AHB clock). This mode can be selected by setting a control bit in the Flash access control register.

Half-cycle access cannot be used when there is a prescaler different from 1 on the AHB clock.

Access latency

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

DCode interface

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

Flash Access controller

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

DCode interface requests have priority over other requests.

4.2.3 Flash program and erase operations

The STM32F302xx embedded Flash memory can be programmed using in-circuit programming or in-application programming.

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

In contrast to the ICP method, **in-application programming (IAP)** can use any communication interface supported by the microcontroller (I/Os, USB, CAN, UART, I²C, SPI, etc.) to download programming data into memory. IAP allows the user to re-program the Flash memory while the application is running. Nevertheless, part of the application has to have been previously programmed in the Flash memory using ICP.

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

- Key register (FLASH_KEYR)
- Option byte key register (FLASH_OPTKEYR)
- Flash control register (FLASH_CR)
- Flash status register (FLASH_SR)
- Flash address register (FLASH_AR)
- Option byte register (FLASH_OBR)
- Write protection register (FLASH_WRP)

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

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

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

Unlocking the Flash memory

After reset, the FPEC is protected against unwanted write or erase operations. The FLASH_CR register is not accessible in write mode, except for the OBL LAUNCH bit, used to reload the OBL. An unlocking sequence should be written to the FLASH_KEYR register to open the access to the FLASH_CR register. This sequence consists of two write operations into FLASH_KEYR register:

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

Any wrong sequence locks up the FPEC and the FLASH_CR register until the next reset.

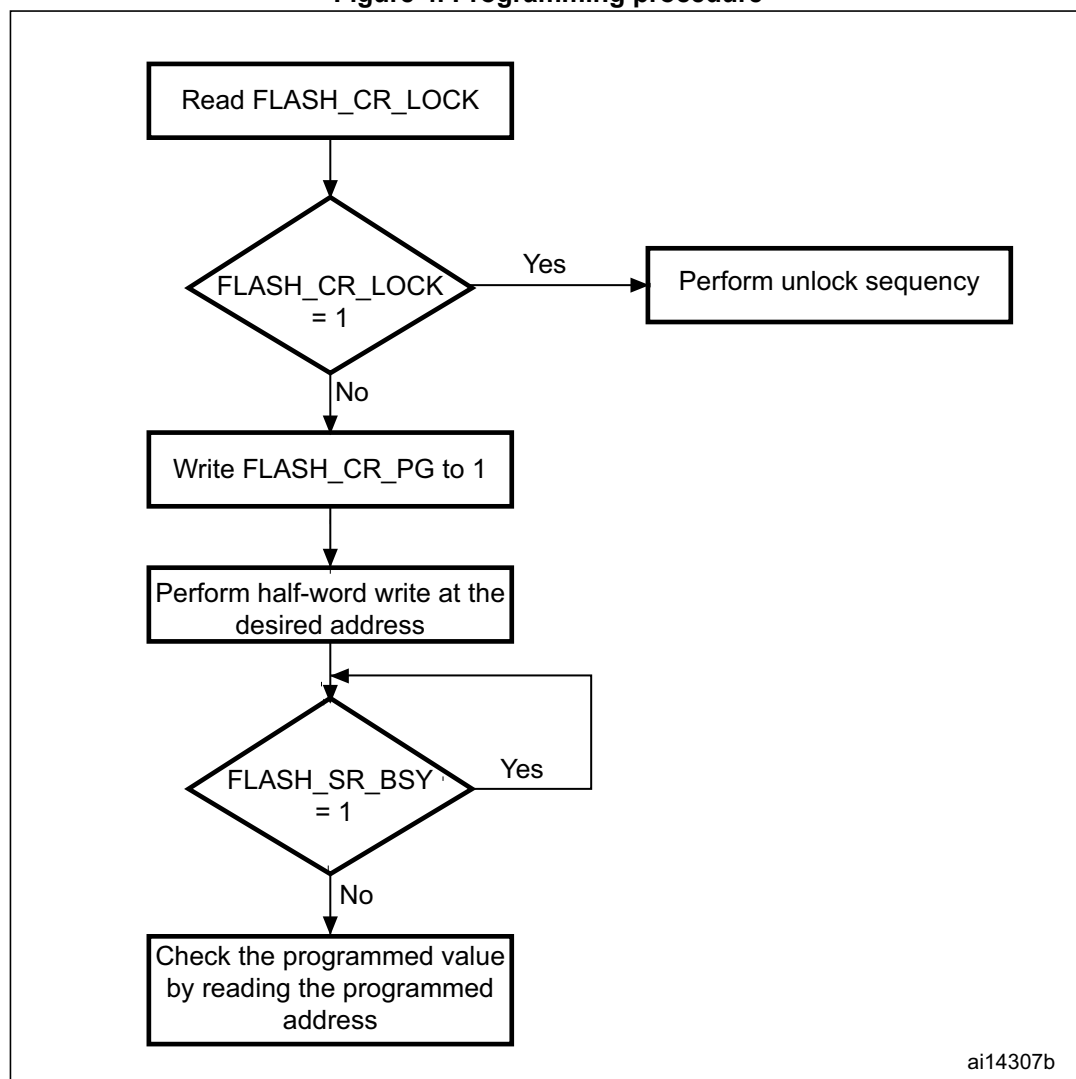
In the case of a wrong key sequence, a bus error is detected and a Hard Fault interrupt is generated. This is done after the first write cycle if KEY1 does not match, or during the second write cycle if KEY1 has been correctly written but KEY2 does not match.

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

Main Flash memory programming

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

Figure 4. Programming procedure



The Flash memory interface preliminarily reads the value at the addressed main Flash memory location and checks that it has been erased. If not, the program operation is skipped and a warning is issued by the PGERR bit in FLASH_SR register (the only exception to this is when 0x0000 is programmed. In this case, the location is correctly

programmed to 0x0000 and the PGERR bit is not set). If the addressed main Flash memory location is write-protected by the FLASH_WRPRT register, the program operation is skipped and a warning is issued by the WRPRTERR bit in the FLASH_SR register. The end of the program operation is indicated by the EOP bit in the FLASH_SR register.

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

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

Note: The registers are not accessible in write mode when the BSY bit of the FLASH_SR register is set.

Flash memory erase

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

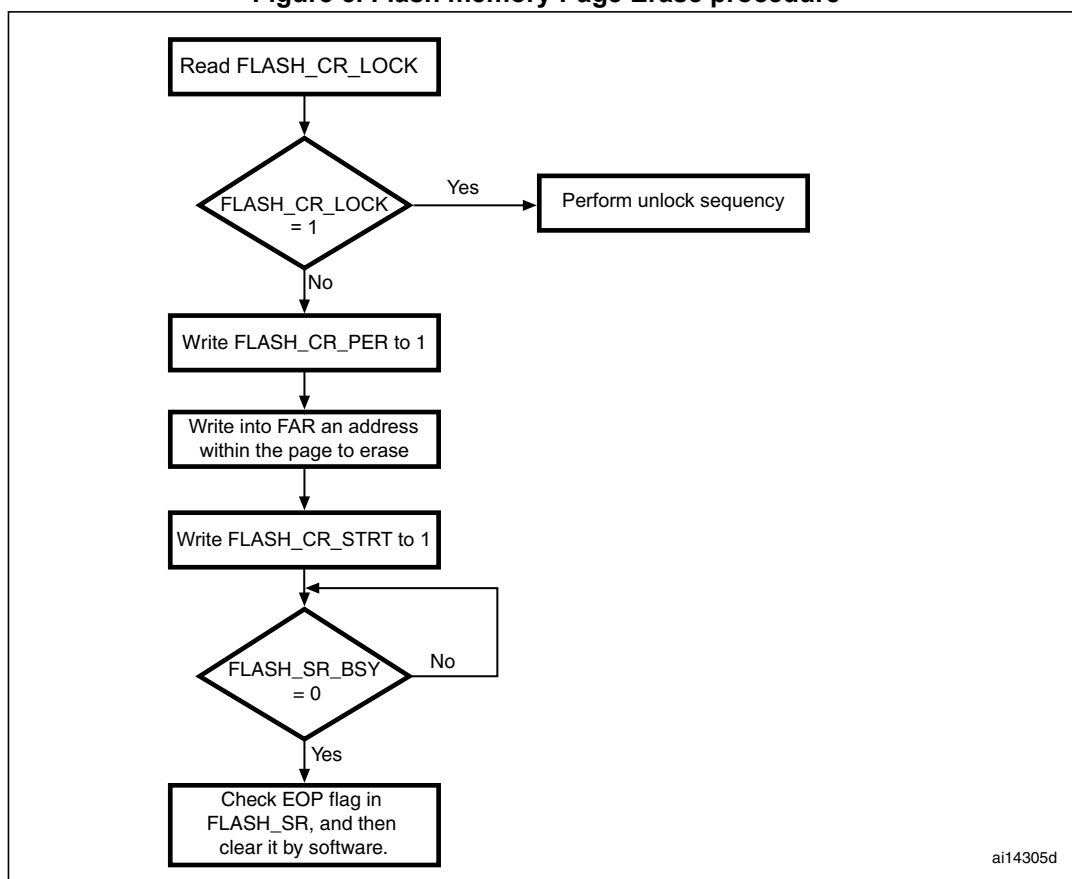
Page Erase

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

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

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

Figure 5. Flash memory Page Erase procedure



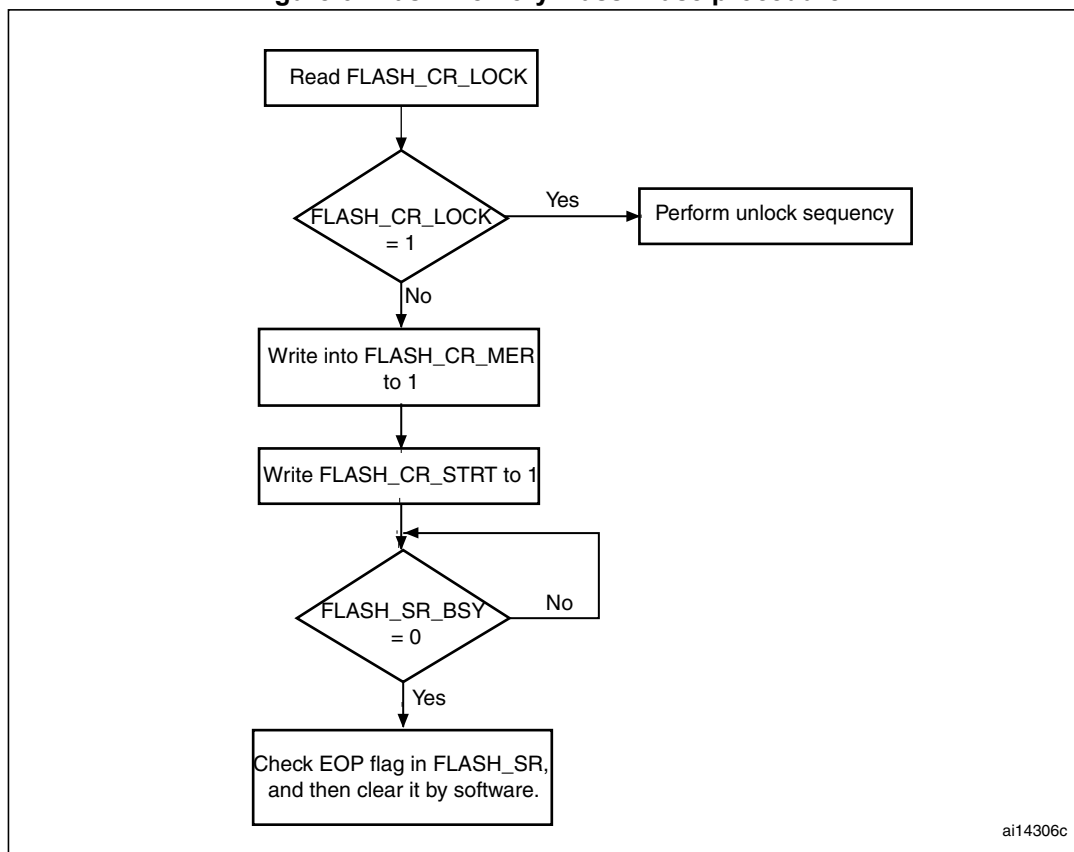
Mass Erase

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

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

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

Figure 6. Flash memory Mass Erase procedure



Option byte programming

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

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

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

The sequence is as follows:

- Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register.
- Unlock the OPTWRE bit in the FLASH_CR register.
- Set the OPTPG bit in the FLASH_CR register
- Write the data (half-word) to the desired address
- Wait for the BSY bit to be reset.
- Read the programmed value and verify.

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

Erase procedure

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

- Check that no Flash memory operation is ongoing by reading the BSY bit in the FLASH_SR register
- Unlock the OPTWRE bit in the FLASH_CR register
- Set the OPTER bit in the FLASH_CR register
- Set the STRT bit in the FLASH_CR register
- Wait for BSY to reset
- Read the erased option bytes and verify

4.3 Memory protection

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

4.3.1 Read protection

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

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

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

The Flash memory is protected when the RDP option byte and its complement contain the pair of values shown in [Table 7](#).

Table 7. Flash memory read protection status

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

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

Level 0: no protection

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

Level 1: Read protection

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

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

Level 2: No debug

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

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

Note: The debug feature is also disabled under reset.

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

Table 8. Access status versus protection level and execution modes

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

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

Changing read protection level

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

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

To validate the protection level change, the option bytes must be reloaded through the OBL_LAUNCH bit in Flash control register.

4.3.2 Write protection

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

If a program or an erase operation is performed on a protected , the Flash memory returns a WRPRTERR protection error flag in the Flash memory Status Register (FLASH_SR).

Write unprotection

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

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

4.3.3 Option byte block write protection

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

4.4 Flash interrupts

Table 9. Flash interrupt request

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

4.5 Flash register description

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

4.5.1 Flash access control register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	PRFT BS	PRFT BE	HLF CYA	LATENCY[2:0]		
										r	rw	rw	rw	rw	rw

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

Bit 5 **PRFTBS**: Prefetch buffer status

This bit provides the status of the prefetch buffer.

0: Prefetch buffer is disabled

1: Prefetch buffer is enabled

Bit 4 **PRFTBE**: Prefetch buffer enable

0: Prefetch is disabled

1: Prefetch is enabled

Bit 3 **HLFCYA**: Flash half cycle access enable

0: Half cycle is disabled

1: Half cycle is enabled

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

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

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

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

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

4.5.2 Flash key register (FLASH_KEYR)

Address offset: 0x04

Reset value: xxxx xxxx

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

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

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

These bits represent the keys to unlock the Flash.

4.5.3 Flash option key register (FLASH_OPTKEYR)

Address offset: 0x08

Reset value: xxxx xxxx

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

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

Bits 31:0 **OPTKEYR**: Option byte key

These bits represent the keys to unlock the OPTWRE.

4.5.4 Flash status register (FLASH_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	EOP	WRPRT ERR	Res.	PG ERR	Res.	BSY
										rw	rw		rw		r

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

Bit 5 **EOP**: End of operation

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

Reset by writing a 1

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

Bit 4 **WRPRTERR**: Write protection error

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

Reset by writing 1.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **PGERR**: Programming error

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

Reset by writing 1.

Note: The STRT bit in the FLASH_CR register should be reset before starting a programming operation.

Bit 1 Reserved, must be kept at reset value

Bit 0 **BSY**: Busy

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

4.5.5 Flash control register (FLASH_CR)

Address offset: 0x10

Reset value: 0x0000 0080

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

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

Bit 13 **OBL_LAUNCH**: Force option byte loading

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

0: Inactive

1: Active

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

This bit enables the interrupt generation when the EOP bit in the FLASH_SR register goes to 1.

0: Interrupt generation disabled

1: Interrupt generation enabled

Bit 11 Reserved, must be kept at reset value

Bit 10 **ERRIE**: Error interrupt enable

This bit enables the interrupt generation on an error when PGERR / WRPRTERR are set in the FLASH_SR register.

0: Interrupt generation disabled

1: Interrupt generation enabled

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

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

This bit can be reset by software

Bit 8 Reserved, must be kept at reset value.

Bit 7 **LOCK**: Lock

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

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

Bit 6 **STRT**: Start

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

Bit 5 **OPTER**: Option byte erase

Option byte erase chosen.

Bit 4 **OPTPG**: Option byte programming

Option byte programming chosen.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **MER**: Mass erase

Erase of all user pages chosen.

Bit 1 **PER**: Page erase

Page Erase chosen.

Bit 0 **PG**: Programming

Flash programming chosen.

4.5.6 Flash address register (FLASH_AR)

Address offset: 0x14

Reset value: 0x0000 0000

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

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

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

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

Note: Write access to this register is blocked when the BSY bit in the FLASH_SR register is set.

4.5.7 Option byte register (FLASH_OBR)

Address offset 0x1C

Reset value: 0xFFFFFFFF

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Data1								Data0								Res.	SRAM_PE	VDDA_MONITOR	nBOOT1	Res.	nRST_STDBY	nRST_STOP	WDG_SW	Res.					RDPRT[1:0]		OPTERR	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r			r	r		r	r	r							r	r	r

Bits 31:24 Data1

Bits 23:16 Data0

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

Bit 15: Reserved, must be kept at reset value.

Bit 14: SRAM_PE

Bit 13: VDDA_MONITOR

Bit 12: nBOOT1

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

Bit 10: nRST_STDBY

Bit 9: nRST_STOP

Bit 8: WDG_SW

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

Bit 2:1 **RDPRT[1:0]**: Read protection Level status

00: Read protection Level 0 is enabled (ST production set up)

01: Read protection Level 1 is enabled

10: Reserved

11: Read protection Level 2 is enabled

Note: These bits are read-only.

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

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

Note: This bit is read-only.

4.5.8 Write protection register (FLASH_WRP)

Address offset: 0x20

Reset value: 0xFFFF FFFF

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

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

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

4.6 Flash register map

Table 10. Flash interface - register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRFTBS	PRFTBE	HLFCYA	LATENCY [2:0]		
	Reset value																											1	1	0	0	0	0
0x004	FLASH_KEYR	FKEYR[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x008	FLASH_OPTKEYR	OPTKEYR[31:0]																															
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x00C	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOP	WRPERR	Res.	PGERR	Res.	BSY
	Reset value																											0	0		0		0
0x010	FLASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OBL_LAUNCH	EOPIE	Res.	ERRIE	OPTWRE	Res.	LOCK	STRT	OPTER	OPTPG	Res.	MER	PER	PG
	Reset value																			0	0		0	0		1	0	0	0		0	0	0
0x014	FLASH_AR	FAR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10. Flash interface - register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x01C	FLASH_OBR	Data1								Data0								Res.	SRAM_PE	VDDA_MONITOR	nBOOT1	Res.	nRST_STDBY	nRST_STOP	WDG_SW	Res.								RDPRT[1:0]		OPTERR
	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	0	0	0	0	x	x	x	x			
0x020	FLASH_WRPR	WRP[31:0]																																		
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

5 Option byte description

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

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

Table 11. Option byte format

31-24	23-16	15 -8	7-0
Complemented option byte1	Option byte 1	Complemented option byte0	Option byte 0

The organization of these bytes inside the information block is as shown in [Table 12](#).

The option bytes can be read from the memory locations listed in [Table 12](#) or from the Option byte register (FLASH_OBR).

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

Table 12. Option byte organization

Address	[31:24]	[23:16]	[15:8]	[7:0]
0x1FFF F800	nUSER	USER	nRDP	RDP
0x1FFF F804	nData1	Data1	nData0	Data0
0x1FFF F808	nWRP1	WRP1	nWRP0	WRP0
0x1FFF F80C	nWRP3 ⁽¹⁾	WRP3 ⁽¹⁾	nWRP2 ⁽¹⁾	WRP2 ⁽¹⁾

1. Available only on STM32F302xB/C/D/E. Must be kept at reset values in STM32F302x6/8 devices.

Table 13. Description of the option bytes

Flash memory address	Option bytes
0x1FFF F800	<p>Bits [31:24]: nUSER</p> <p>Bits [23:16]: USER: User option byte (stored in FLASH_OBR[15:8])</p> <p>This byte is used to configure the following features:</p> <ul style="list-style-type: none"> - Select the watchdog event: Hardware or software - Reset event when entering Stop mode - Reset event when entering Standby mode <p>Bit 23: Reserved</p> <p>Bit 22: SRAM_PE (not available in STM32F302x6/8 devices)</p> <p>The SRAM hardware parity check is disabled by default. This bit allows the user to enable the SRAM hardware parity check.</p> <ul style="list-style-type: none"> 0: Parity check enabled. 1: Parity check disabled. <p>Bit 21: VDDA_MONITOR</p> <p>This bit selects the analog monitoring on the VDDA power source:</p> <ul style="list-style-type: none"> 0: VDDA power supply supervisor disabled. 1: VDDA power supply supervisor enabled. <p>Bit 20: nBOOT1</p> <p>Together with the BOOT0 pin, this bit selects Boot mode from the main Flash memory, SRAM or System memory. Refer to Section 3.5 on page 57.</p> <p>Bit 19: Reserved, must be kept at reset.</p> <p>Bit 18: nRST_STDBY</p> <ul style="list-style-type: none"> 0: Reset generated when entering Standby mode. 1: No reset generated. <p>Bit 17: nRST_STOP</p> <ul style="list-style-type: none"> 0: Reset generated when entering Stop mode 1: No reset generated <p>Bit 16: WDG_SW</p> <ul style="list-style-type: none"> 0: Hardware watchdog 1: Software watchdog <p>Bits [15:8]: nRDP</p> <p>Bits [7:0]: RDPR: Read protection option byte</p> <p>The value of this byte defines the Flash memory protection level</p> <ul style="list-style-type: none"> 0xAA: Level 0 0xFF (except 0xAA and 0xCC): Level 1 0xCC: Level 2 <p>The protection levels are stored in the Flash_OBR Flash option bytes register (RDPRT bits).</p>
0x1FFF F804	<p>Dat: Two bytes for user data storage.</p> <p>These addresses can be programmed using the option byte programming procedure.</p> <p>Bits [31:24]: nData1</p> <p>Bits [23:16]: Data1 (stored in FLASH_OBR[31:24])</p> <p>Bits [15:8]: nData0</p> <p>Bits [7:0]: Data0 (stored in FLASH_OBR[23:16])</p>

Table 13. Description of the option bytes (continued)

Flash memory address	Option bytes
0x1FFF F808	<p>WRP_x: Flash memory write protection option bytes</p> <p>Bits [31:24]: nWRP1</p> <p>Bits [23:16]: WRP1 (stored in FLASH_WRP[15:8])</p> <p>Bits [15:8]: nWRP0</p> <p>Bits [7:0]: WRP0 (stored in FLASH_WRP[7:0])</p> <p>0: Write protection active</p> <p>1: Write protection not active</p> <p>Refer to Section 4.3.2: Write protection for more details.</p> <p><i>Note: Even if WRP2 and WRP3 are not available, they must be kept at reset value.</i></p>
0x1FFF F80C	<p>WRP_x: Flash memory write protection option bytes (available only on STM32F302xB/C)</p> <p>Bits [31:24]: nWRP3</p> <p>Bits [23:16]: WRP3 (stored in FLASH_WRP[31:24])</p> <p>Bits [15:8]: nWRP2</p> <p>Bits [7:0]: WRP2 (stored in FLASH_WRP[23:16])</p> <p>One bit of the user option bytes WRP_x is used to protect 2 pages of 2 Kbytes in the main memory block.</p> <p>0: Write protection active</p> <p>1: Write protection not active</p> <p>In total, 4 user option bytes are used to protect the whole main Flash memory.</p> <p>WRP0: Write-protects pages 0 to 15</p> <p>WRP1: Write-protects pages 16 to 31</p> <p>WRP2: Write-protects pages 32 to 47⁽¹⁾</p> <p>WRP3: bits 0-6 write-protect pages 48 to 61, bit 7 write-protects pages 62 to 127 on STM32F302xB/C, pages 63 to 255 on STM32F302xD/E⁽¹⁾</p>

1. Even if WRP2 and WRP3 are not available on the STM32F302x6/8, they must not be re-programmed and must be kept at reset value 0xFF.

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

6 Cyclic redundancy check calculation unit (CRC)

6.1 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.

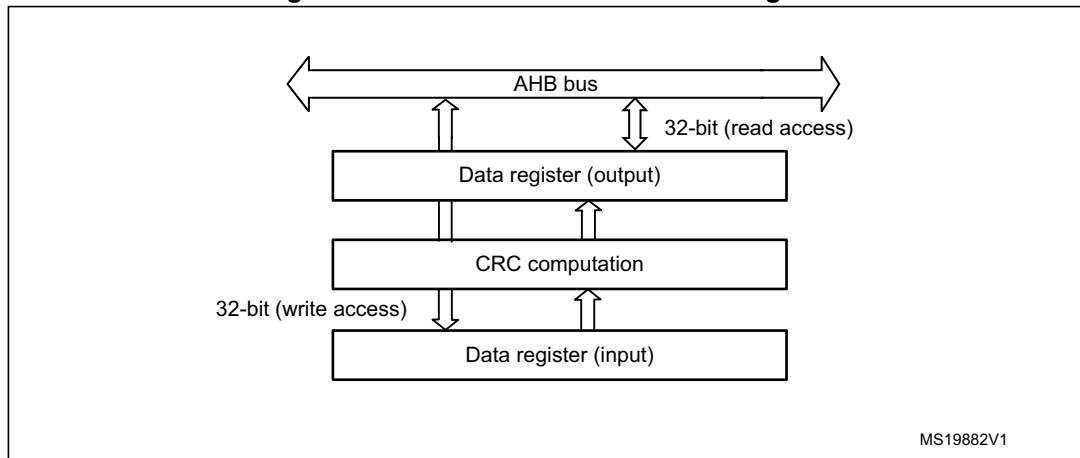
Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to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

6.2 CRC main features

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

6.3 CRC functional description

Figure 7. CRC calculation unit block diagram



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

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

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

The duration of the computation depends on data width:

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

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

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

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

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

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

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

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

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

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

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

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. The polynomial must be configured with an odd value. Even values are not supported.

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.

6.4 CRC registers

6.4.1 Data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

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

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

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

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

6.4.2 Independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	IDR[7:0]							
								rw							

Bits 31:8 Reserved, must be kept cleared.

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

These bits can be used as a temporary storage location for one byte.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

6.4.3 Control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bit 7 **REV_OUT**: Reverse output data

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

0: Bit order not affected

1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data

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

00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word

11: Bit reversal done by word

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

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept cleared.

Bit 0 **RESET**: RESET bit

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

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

Bits 31:0 **CRC_INIT**: *Programmable initial CRC value*

This register is used to write the CRC initial value.

6.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C11DB7

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

6.4.6 CRC register map

Table 14. CRC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IDR[7:0]							
	Reset value																									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	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res	Res	RESET
	Reset value																									0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	Polynomial coefficients																															
	Reset value	0x04C11DB7																															

Refer to [Section 3.2.2 on page 48](#) for the register boundary addresses.

7 Flexible memory controller (FMC)

Note: Only the STM32F302xD/E devices include the FMC.

The Flexible memory controller (FMC) includes two memory controllers:

- The NOR/PSRAM memory controller
- The NAND/PC Card memory controller

7.1 FMC main features

The FMC functional block makes the interface with: synchronous and asynchronous static memories, and 16-bit PC card memory. Its main purposes are:

- to translate AHB transactions into the appropriate external device protocol
- to meet the access time requirements of the external memory devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique Chip Select. The FMC performs only one access at a time to an external device.

The main features of the FMC controller are the following:

- Interface with static-memory mapped devices including:
 - Static random access memory (SRAM)
 - NOR Flash memory/OneNAND Flash memory
 - PSRAM (4 memory banks)
 - 16-bit PC Card compatible devices
 - Two banks of NAND Flash memory with ECC hardware to check up to 8 Kbytes of data
- Burst mode support for faster access to synchronous devices such as NOR Flash memory, PSRAM
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-,16-bit wide data bus
- Independent Chip Select control for each memory bank
- Independent configuration for each memory bank
- Write enable and byte lane select outputs for use with PSRAM, SRAM devices
- External asynchronous wait control
- Write Data FIFO with 16 x33-bit depth
- Write Address FIFO with 16x30-bit depth

The FMC embeds two Write FIFOs: a Write Data FIFO with a 16x33-bit depth and a Write Address FIFO with a 16x30-bit depth.

- The Write Data FIFO stores the AHB data to be written to the memory (up to 32 bits) plus one bit for the AHB transfer (burst or not sequential mode)
- The Write Address FIFO stores the AHB address (up to 28 bits) plus the AHB data size (up to 2 bits). When operating in burst mode, only the start address is stored except when crossing a page boundary (for PSRAM). In this case, the AHB burst is broken into two FIFO entries.

At startup the FMC pins must be configured by the user application. The FMC I/O pins which are not used by the application can be used for other purposes.

The FMC registers that define the external device type and associated characteristics are usually set at boot time and do not change until the next reset or power-up. However, the settings can be changed at any time.

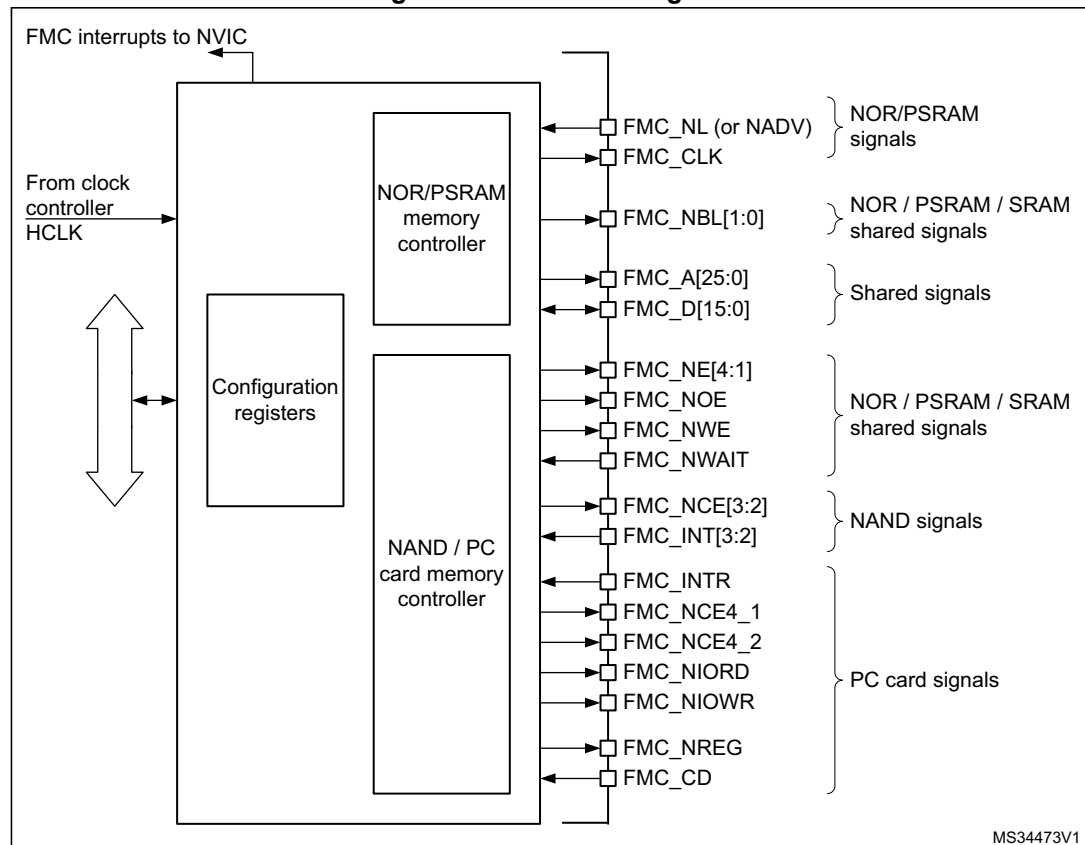
7.2 Block diagram

The FMC consists of the following main blocks:

- The AHB interface (including the FMC configuration registers)
- The NOR Flash/PSRAM/SRAM controller
- The NAND Flash/PC Card controller
- The external device interface

The block diagram is shown in *Figure 8*.

Figure 8. FMC block diagram



7.3 AHB interface

The AHB slave interface allows internal CPUs and other bus master peripherals to access the external memories.

AHB transactions are translated into the external device protocol. In particular, if the selected external memory is 16- or 8-bit wide, 32-bit wide transactions on the AHB are split into consecutive 16- or 8-bit accesses. The FMC Chip Select (FMC_NEx) does not toggle between the consecutive accesses.

The FMC generates an AHB error in the following conditions:

- When reading or writing to an FMC bank (Bank 1 to 4) which is not enabled.
- When reading or writing to the NOR Flash bank while the FACCEN bit is reset in the FMC_BCRx register.
- When reading or writing to the PC Card banks while the FMC_CD input pin (Card Presence Detection) is low.

The effect of an AHB error depends on the AHB master which has attempted the R/W access:

- If the access has been attempted by the Cortex™-M4 with FPU CPU, a hard fault interrupt is generated.
- If the access has been performed by a DMA controller, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

The AHB clock (HCLK) is the reference clock for the FMC.

7.3.1 Supported memories and transactions

General transaction rules

The requested AHB transaction data size can be 8-, 16- or 32-bit wide whereas the accessed external device has a fixed data width. This may lead to inconsistent transfers.

Therefore, some simple transaction rules must be followed:

- AHB transaction size and memory data size are equal
There is no issue in this case.
- AHB transaction size is greater than the memory size:
In this case, the FMC splits the AHB transaction into smaller consecutive memory accesses to meet the external data width. The FMC Chip Select (FMC_NEx) does not toggle between the consecutive accesses.
- AHB transaction size is smaller than the memory size:
The transfer may or not be consistent depending on the type of external device:
 - Accesses to devices that have the byte select feature (SRAM, ROM, PSRAM)
In this case, the FMC allows read/write transactions and accesses the right data through its byte lanes NBL[1:0].
Bytes to be written are addressed by NBL[1:0].
All memory bytes are read (NBL[1:0] are driven low during read transaction) and the useless ones are discarded.
 - Accesses to devices that do not have the byte select feature (NOR and NAND Flash memories)
This situation occurs when a byte access is requested to a 16-bit wide Flash memory. Since the device cannot be accessed in byte mode (only 16-bit words can be read/written from/to the Flash memory), Write transactions and Read transactions are allowed (the controller reads the entire 16-bit memory word and uses only the required byte).

Configuration registers

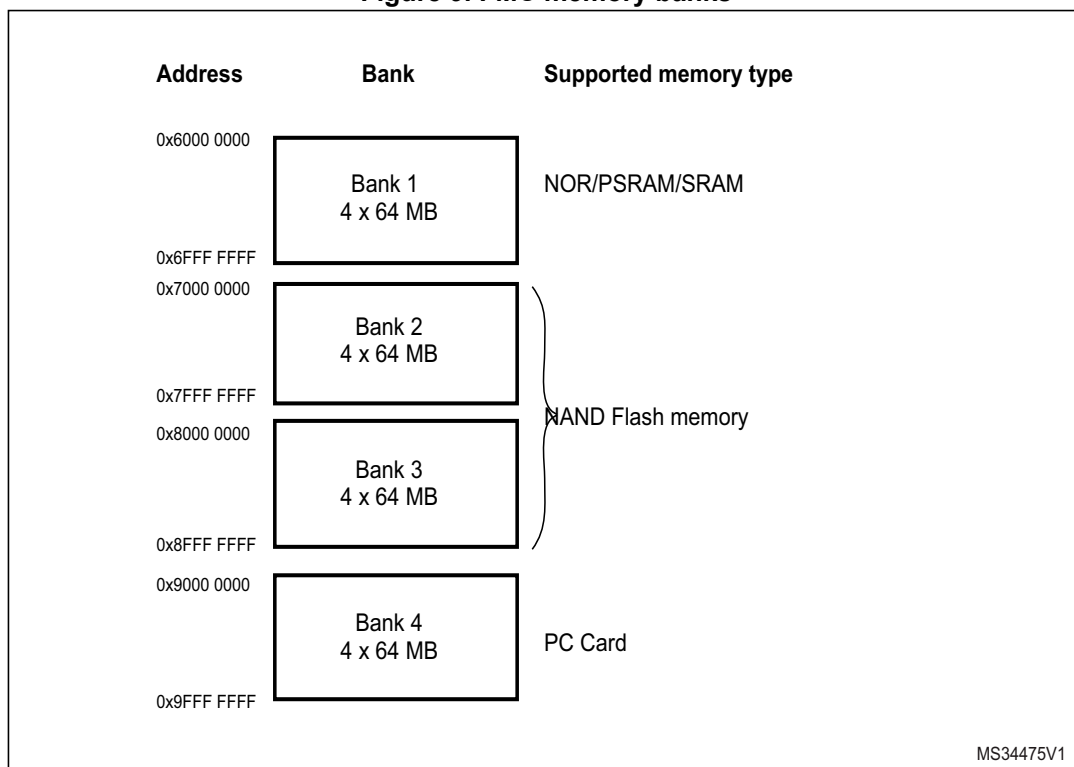
The FMC can be configured through a set of registers. Refer to [Section 7.5.6](#), for a detailed description of the NOR Flash/PSRAM controller registers. Refer to [Section 7.6.8](#), for a detailed description of the NAND Flash/PC Card registers.

7.4 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size banks of 256 Mbytes each (see [Figure 9](#)):

- Bank 1 used to address up to 4 NOR Flash memory or PSRAM devices. This bank is split into 4 NOR/PSRAM subbanks with 4 dedicated Chip Selects, as follows:
 - Bank 1 - NOR/PSRAM 1
 - Bank 1 - NOR/PSRAM 2
 - Bank 1 - NOR/PSRAM 3
 - Bank 1 - NOR/PSRAM 4
- Banks 2 and 3 used to address NAND Flash memory devices (1 device per bank)
- Bank 4 used to address a PC Card

For each bank the type of memory to be used can be configured by the user application through the Configuration register.

Figure 9. FMC memory banks

7.4.1 NOR/PSRAM address mapping

HADDR[27:26] bits are used to select one of the four memory banks as shown in [Table 15](#).

Table 15. NOR/PSRAM bank selection

HADDR[27:26] ⁽¹⁾	Selected bank
00	Bank 1 - NOR/PSRAM 1
01	Bank 1 - NOR/PSRAM 2
10	Bank 1 - NOR/PSRAM 3
11	Bank 1 - NOR/PSRAM 4

1. HADDR are internal AHB address lines that are translated to external memory.

The HADDR[25:0] bits contain the external memory address. Since HADDR is a byte address whereas the memory is addressed at word level, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

Table 16. NOR/PSRAM External memory address

Memory width ⁽¹⁾	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	HADDR[25:0]	64 Mbytes x 8 = 512 Mbit
16-bit	HADDR[25:1] >> 1	64 Mbytes/2 x 16 = 512 Mbit

1. In case of a 16-bit external memory width, the FMC will internally use HADDR[25:1] to generate the address for external memory FMC_A[24:0].
Whatever the external memory width, FMC_A[0] should be connected to external memory address A[0].

Wrap support for NOR Flash/PSRAM

Wrap burst mode for synchronous memories is not supported. The memories must be configured in linear burst mode of undefined length.

7.4.2 NAND Flash memory/PC Card address mapping

In this case, three banks are available, each of them being divided into memory areas as indicated in [Table 17](#).

Table 17. NAND/PC Card memory mapping and timing registers

Start address	End address	FMC bank	Memory space	Timing register
0x9C00 0000	0x9FFF FFFF	Bank 4 - PC card	I/O	FMC_PIO4 (0xB0)
0x9800 0000	0x9BFF FFFF		Attribute	FMC_PATT4 (0xAC)
0x9000 0000	0x93FF FFFF		Common	FMC_PMEM4 (0xA8)
0x8800 0000	0x8BFF FFFF	Bank 3 - NAND Flash	Attribute	FMC_PATT3 (0x8C)
0x8000 0000	0x83FF FFFF		Common	FMC_PMEM3 (0x88)
0x7800 0000	0x7BFF FFFF	Bank 2- NAND Flash	Attribute	FMC_PATT2 (0x6C)
0x7000 0000	0x73FF FFFF		Common	FMC_PMEM2 (0x68)

For NAND Flash memory, the common and attribute memory spaces are subdivided into three sections (see in [Table 18](#) below) located in the lower 256 Kbytes:

- Data section (first 64 Kbytes in the common/attribute memory space)
- Command section (second 64 Kbytes in the common / attribute memory space)
- Address section (next 128 Kbytes in the common / attribute memory space)

Table 18. NAND bank selection

Section name	HADDR[17:16]	Address range
Address section	1X	0x020000-0x03FFFF
Command section	01	0x010000-0x01FFFF
Data section	00	0x000000-0x0FFFFF

The application software uses the 3 sections to access the NAND Flash memory:

- **To sending a command to NAND Flash memory**, the software must write the command value to any memory location in the command section.
- **To specify the NAND Flash address that must be read or written**, the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several

consecutive write operations to the address section are required to specify the full address.

- **To read or write data**, the software reads or writes the data from/to any memory location in the data section.

Since the NAND Flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

7.5 NOR Flash/PSRAM controller

The FMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM and ROM
 - 8 bits
 - 16 bits
- PSRAM (Cellular RAM)
 - Asynchronous mode
 - Burst mode for synchronous accesses
 - Multiplexed or non-multiplexed
- NOR Flash memory
 - Asynchronous mode
 - Burst mode for synchronous accesses
 - Multiplexed or non-multiplexed

The FMC outputs a unique Chip Select signal, NE[4:1], per bank. All the other signals (addresses, data and control) are shared.

The FMC supports a wide range of devices through a programmable timings among which:

- Programmable wait states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock (FMC_CLK) output.

The FMC Clock (FMC_CLK) is a submultiple of the HCLK clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FMC_BCR1 register:

- If the CCKEN bit is reset, the FMC generates the clock (CLK) only during synchronous accesses (Read/write transactions).
- If the CCKEN bit is set, the FMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FMC_CLK continuous clock, Bank 1 must be configured in synchronous mode (see [Section 7.5.6: NOR/PSRAM controller registers](#)). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AHB data size has to be the same as the memory data width (MWID) otherwise the FMC_CLK frequency will be changed depending on AHB data transaction (refer to [Section 7.5.5: Synchronous transactions](#) for FMC_CLK divider ratio formula).

The size of each bank is fixed and equal to 64 Mbytes. Each bank is configured through dedicated registers (see [Section 7.5.6: NOR/PSRAM controller registers](#)).

The programmable memory parameters include access times (see [Table 19](#)) and support for wait management (for PSRAM and NOR Flash accessed in burst mode).

Table 19. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	AHB clock cycle (HCLK)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	AHB clock cycle (HCLK)	1	15
Data setup	Duration of the data setup phase	Asynchronous	AHB clock cycle (HCLK)	1	256
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read/write	AHB clock cycle (HCLK)	0	15
Clock divide ratio	Number of AHB clock cycles (HCLK) to build one memory clock cycle (CLK)	Synchronous	AHB clock cycle (HCLK)	2	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (CLK)	2	17

7.5.1 External memory interface signals

[Table 20](#), [Table 21](#) and [Table 22](#) list the signals that are typically used to interface with NOR Flash memory, SRAM and PSRAM.

Note: The prefix “N” identifies the signals which are active low.

NOR Flash memory, non-multiplexed I/Os

Table 20. Non-multiplexed I/O NOR Flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip Select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FMC

The maximum capacity is 512 Mbits (26 address lines).

NOR Flash memory, 16-bit multiplexed I/Os**Table 21. 16-bit multiplexed I/O NOR Flash memory**

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FMC

The maximum capacity is 512 Mbits.

PSRAM/SRAM, non-multiplexed I/Os**Table 22. Non-multiplexed I/Os PSRAM/SRAM**

FMC signal name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Data bidirectional bus
NE[x]	O	Chip Select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)

The maximum capacity is 512 Mbits.

PSRAM, 16-bit multiplexed I/Os**Table 23. 16-Bit multiplexed I/O PSRAM**

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus

Table 23. 16-Bit multiplexed I/O PSRAM (continued)

FMC signal name	I/O	Function
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)

The maximum capacity is 512 Mbits (26 address lines).

7.5.2 Supported memories and transactions

[Table 24](#) below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR Flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FMC are shown in gray in this example.

Table 24. NOR Flash/PSRAM: Example of supported memories and transactions

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
NOR Flash (muxed I/Os and nonmuxed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	N	
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32	16	Y	

Table 24. NOR Flash/PSRAM: Example of supported memories and transactions

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
PSRAM (multiplexed I/Os and non-multiplexed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32	16	Y	
	Synchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Synchronous	W	16/32	16	Y	
SRAM and ROM	Asynchronous	R	8 / 16	16	Y	
	Asynchronous	W	8 / 16	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses Use of byte lanes NBL[1:0]

7.5.3 General timing rules

Signals synchronization

- All controller output signals change on the rising edge of the internal clock (HCLK)
- In synchronous mode (read or write), all output signals change on the rising edge of HCLK. Whatever the CLKDIV value, all outputs change as follows:
 - NOEL/NWEL/ NEL/NADVL/ NADVH/ NBLL/ Address valid outputs change on the falling edge of FMC_CLK clock.
 - NOEH/ NWEH/ NEH/ NOEH/NBLH/ Address invalid outputs change on the rising edge of FMC_CLK clock.

7.5.4 NOR Flash/PSRAM controller asynchronous transactions

Asynchronous static memories (NOR Flash, PSRAM, SRAM)

- Signals are synchronized by the internal clock HCLK. This clock is not issued to the memory
- The FMC always samples the data before de-asserting the Chip Select signal NE. This guarantees that the memory data hold timing constraint is met (minimum Chip Enable high to data transition is usually 0 ns)
- If the extended mode is enabled (EXTMOD bit is set in the FMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B.
- If the extended mode is disabled (EXTMOD bit is reset in the FMC_BCRx register), the FMC can operate in Mode1 or Mode2 as follows:
 - Mode 1 is the default mode when SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01 in the FMC_BCRx register)
 - Mode 2 is the default mode when NOR memory type is selected (MTYP = 0x10 in the FMC_BCRx register).

Mode 1 - SRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FMC_BCRx, and FMC_BTRx/FMC_BWTRx registers.

Figure 10. Mode1 read access waveforms

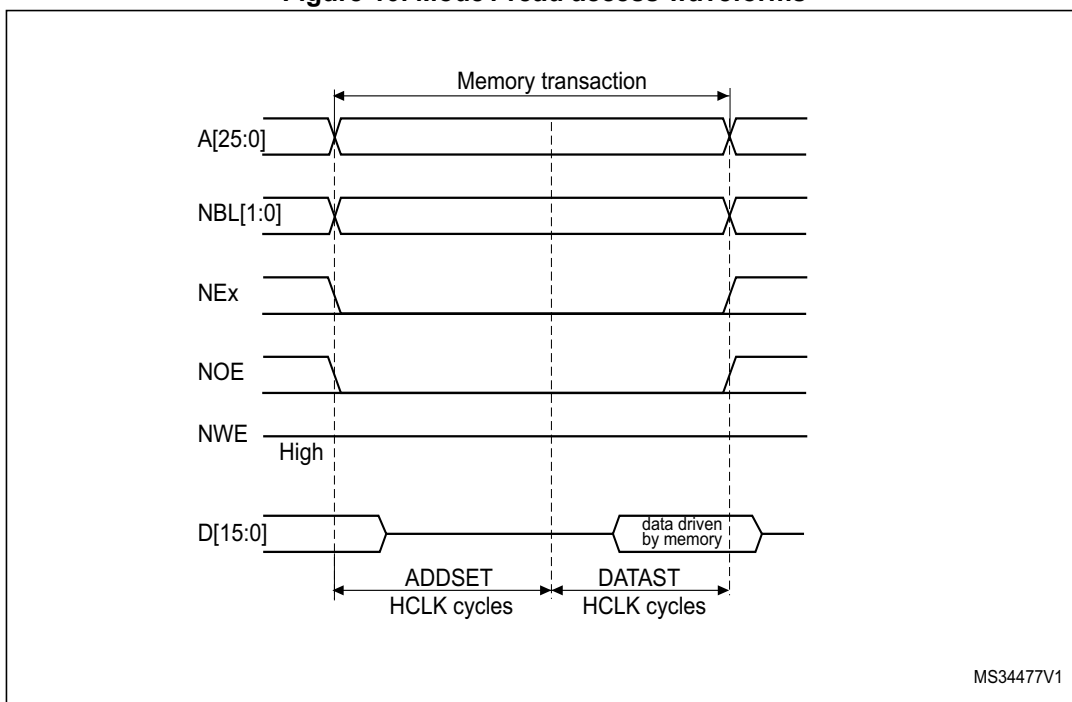
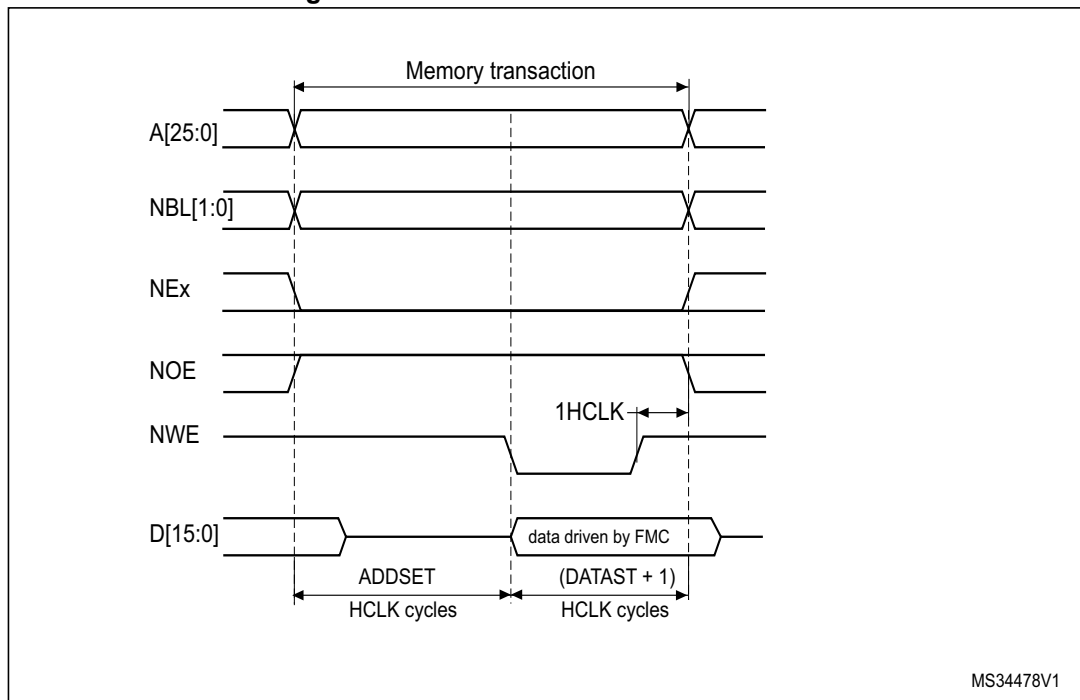


Figure 11. Mode1 write access waveforms



The one HCLK cycle at the end of the write transaction helps guarantee the address and data hold time after the NWE rising edge. Due to the presence of this HCLK cycle, the DATAST value must be greater than zero ($\text{DATAST} > 0$).

Table 25. FMC_BCRx bit fields

Bit number	Bit name	Value to set
31-21	Reserved	0x000
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	Reserved	0x0
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don't care

Table 25. FMC_BCRx bit fields (continued)

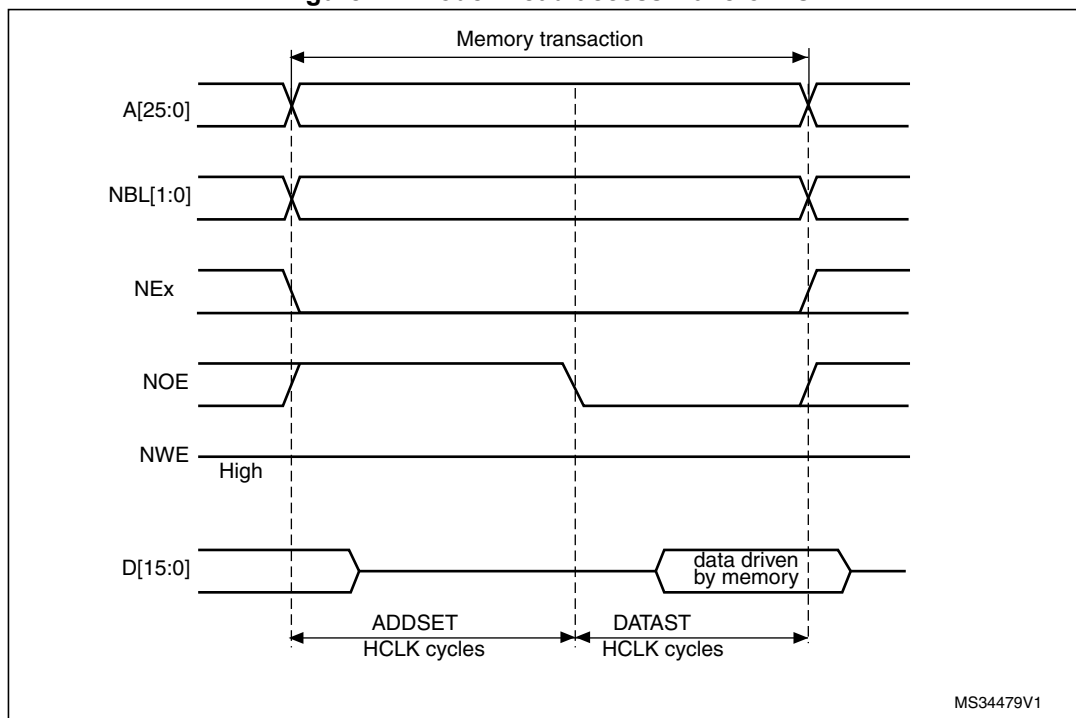
Bit number	Bit name	Value to set
5-4	MWID	As needed
3-2	MTYP	As needed, exclude 0x2 (NOR Flash memory)
1	MUXE	0x0
0	MBKEN	0x1

Table 26. FMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	Don't care
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles for write accesses, DATAST HCLK cycles for read accesses).
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 0.

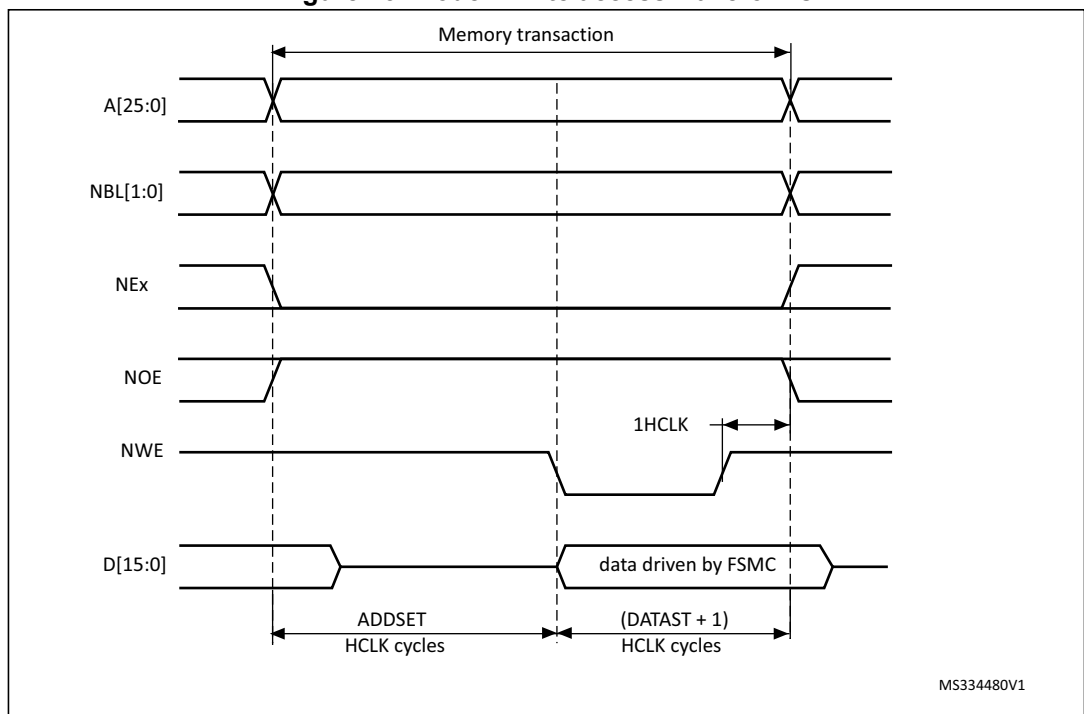
Mode A - SRAM/PSRAM (CRAM) OE toggling

Figure 12. ModeA read access waveforms



1. NBL[1:0] are driven low during the read access

Figure 13. ModeA write access waveforms



The differences compared with mode1 are the toggling of NOE and the independent read and write timings.

Table 27. FMC_BCRx bit fields

Bit number	Bit name	Value to set
31-21	Reserved	0x000
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	Reserved	0x0
15	ASYNCAWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don't care
5-4	MWID	As needed
3-2	MTYP	As needed, exclude 0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 28. FMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 29. FMC_BWTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for write accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Mode 2/B - NOR Flash

Figure 14. Mode2 and mode B read access waveforms

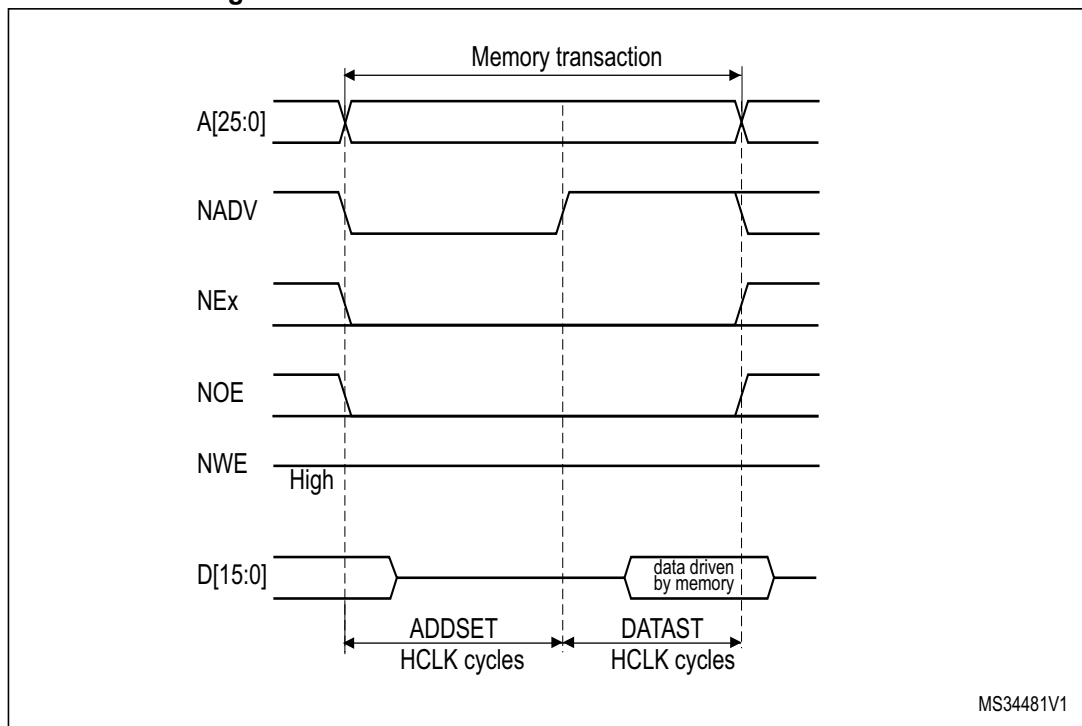


Figure 15. Mode2 write access waveforms

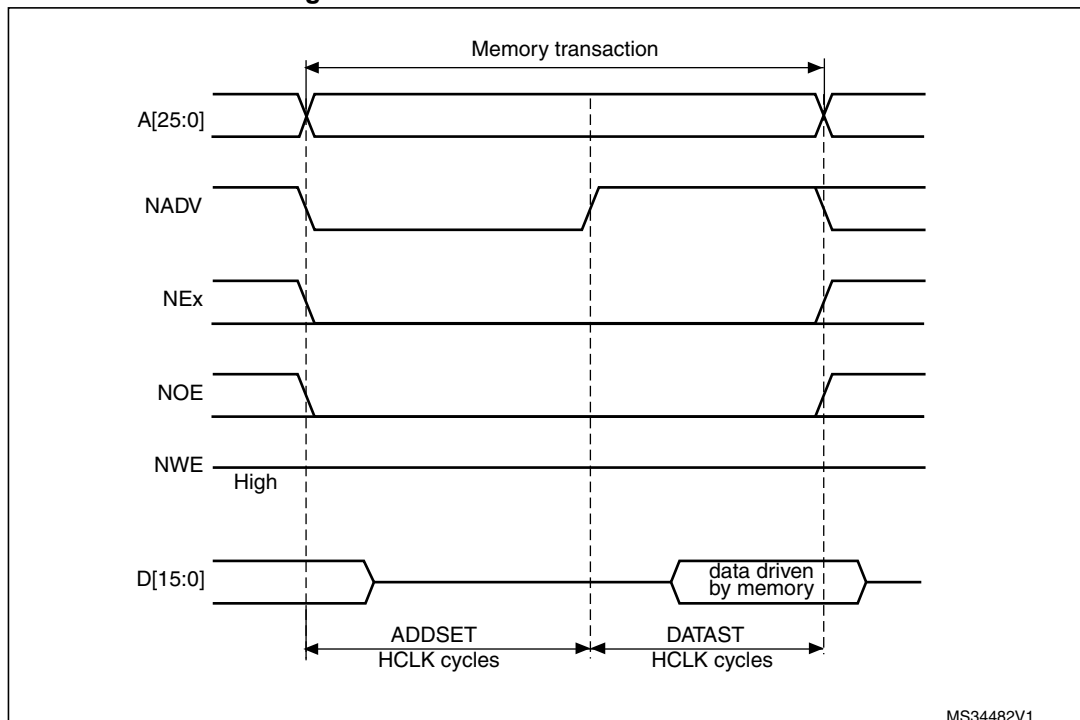
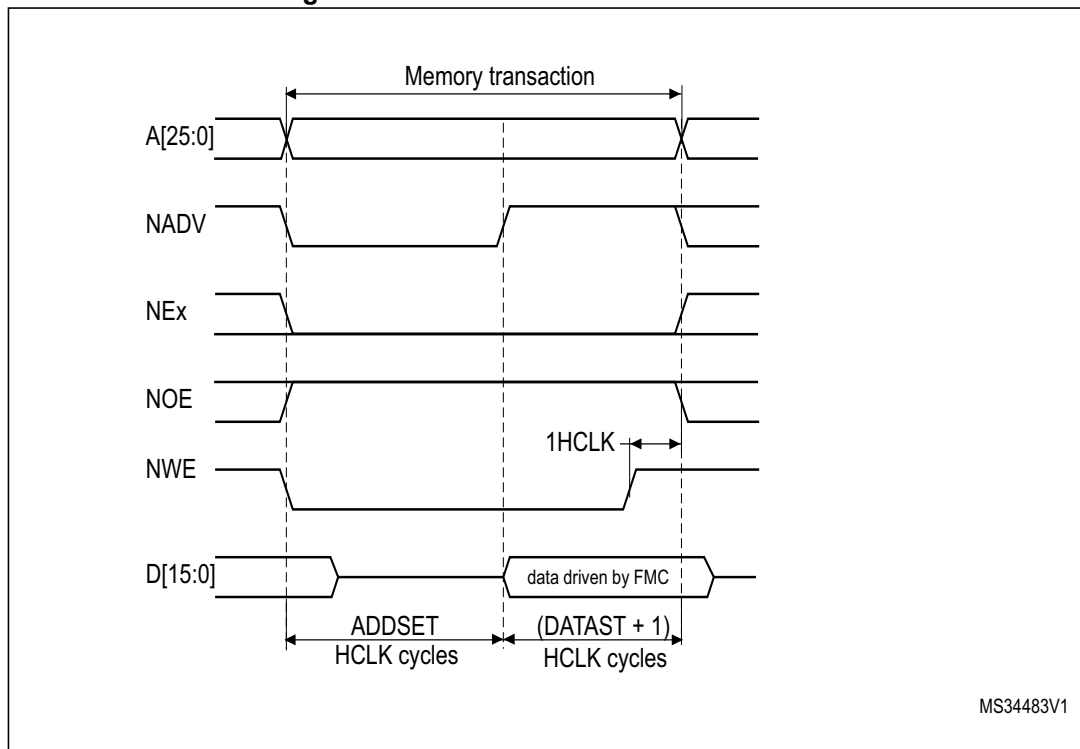


Figure 16. ModeB write access waveforms



The differences with mode1 are the toggling of NWE and the independent read and write timings when extended mode is set (Mode B).

Table 30. FMC_BCRx bit fields

Bit number	Bit name	Value to set
31-21	Reserved	0x000
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	Reserved	0x0
15	ASYNCAWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 31. FMC_BTRx bit fields

Bit number	Bit name	Value to set
31-30	Reserved	0x0
29-28	ACCMOD	0x1 if extended mode is set
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the access second phase (DATAST HCLK cycles) for read accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 32. FMC_BWTRx bit fields

Bit number	Bit name	Value to set
31-30	Reserved	0x0
29-28	ACCMOD	0x1 if extended mode is set
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the access second phase (DATAST HCLK cycles) for write accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Note: The FMC_BWTRx register is valid only if the extended mode is set (mode B), otherwise its content is don't care.

Mode C - NOR Flash - OE toggling

Figure 17. ModeC read access waveforms

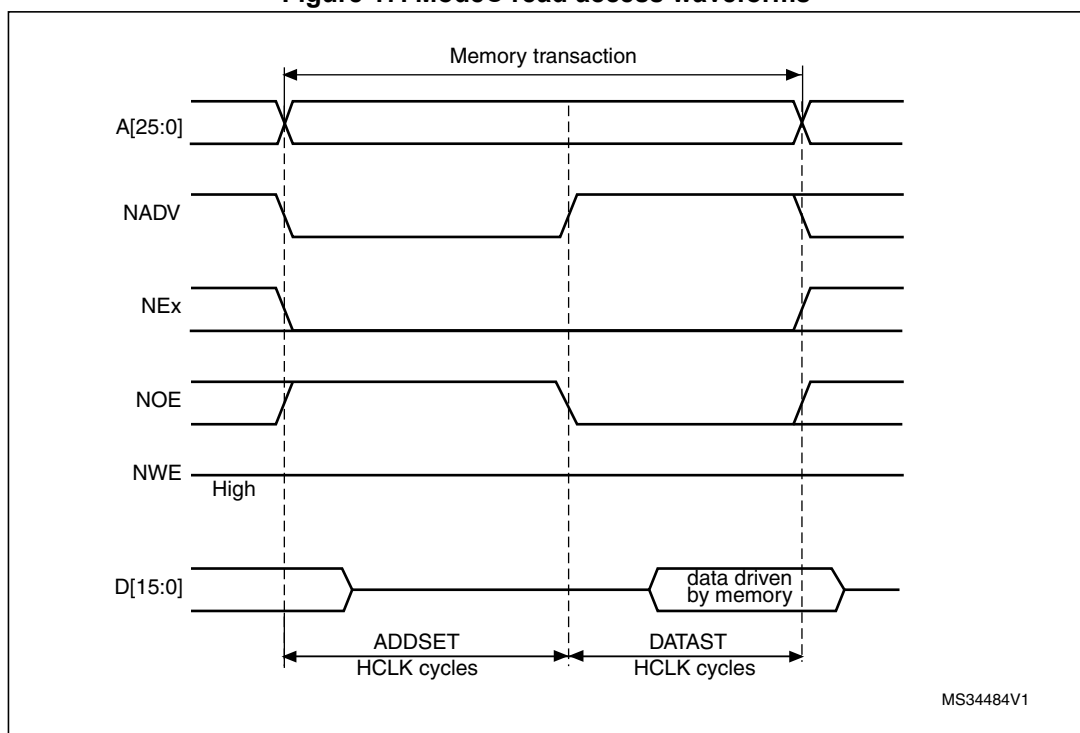
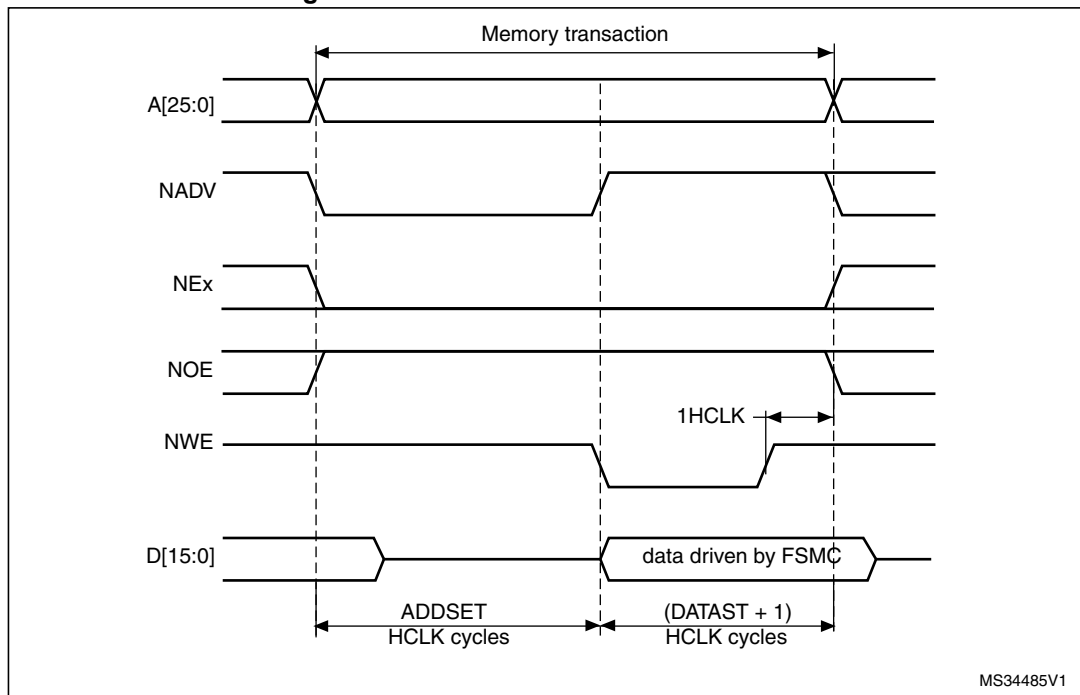


Figure 18. ModeC write access waveforms



MS34485V1

The differences compared with mode1 are the toggling of NOE and the independent read and write timings.

Table 33. FMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-21	Reserved	0x000
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	Reserved	0x0
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	0x02 (NOR Flash memory)

Table 33. FMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
1	MUXEN	0x0
0	MBKEN	0x1

Table 34. FMC_BTRx bit fields

Bit No.	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 35. FMC_BWTRx bit fields

Bit No.	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for write accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Mode D - asynchronous access with extended address

Figure 19. ModeD read access waveforms

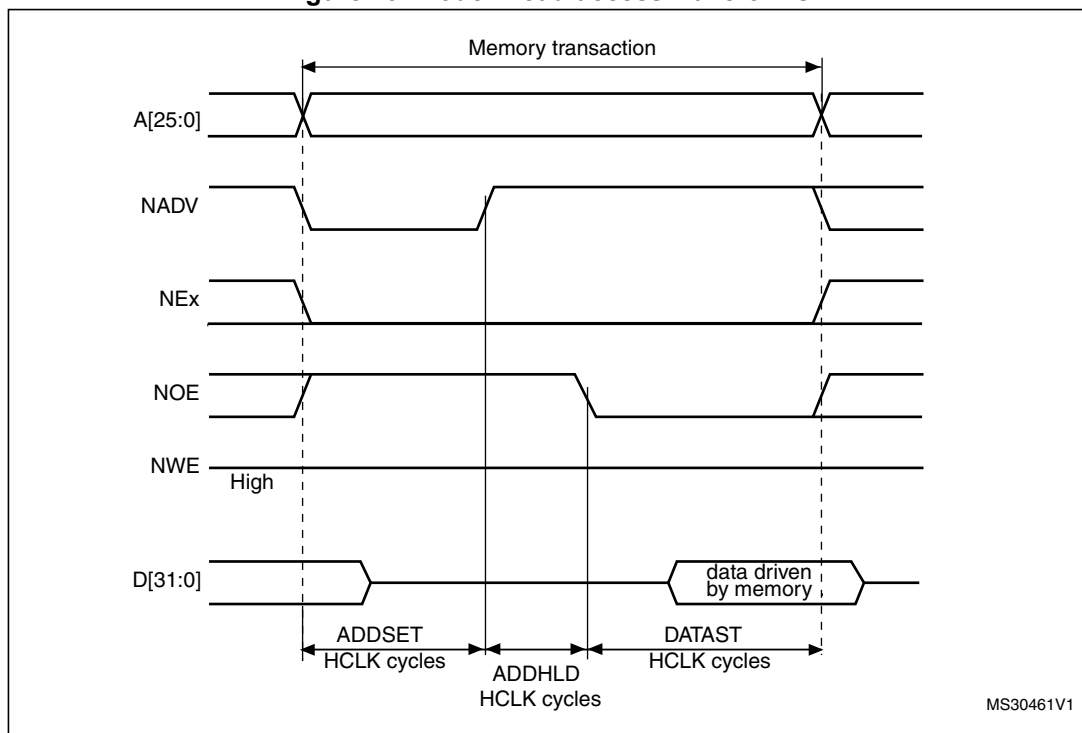
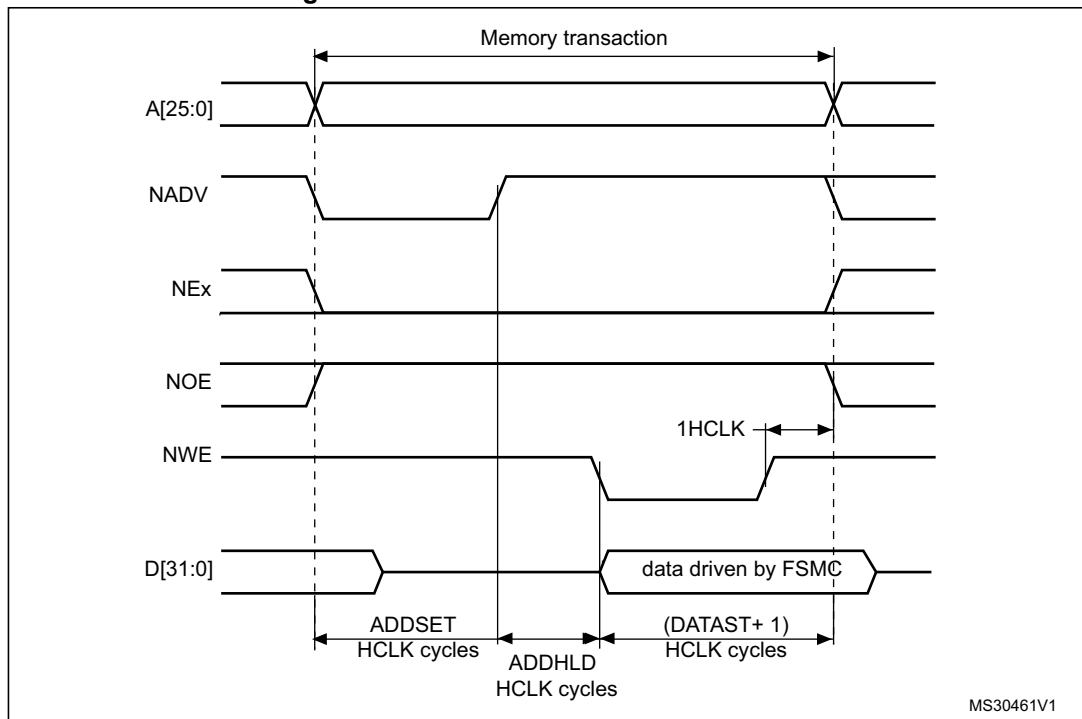


Figure 20. ModeD write access waveforms



The differences with mode1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 36. FMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-21	Reserved	0x000
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	Reserved	0x0
15	ASYNCAWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 37. FMC_BTRx bit fields

Bit No.	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7-4	ADDHLD	Duration of the middle phase of the read access (ADDHLD HCLK cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 1.

Table 38. FMC_BWTRx bit fields

Bit No.	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST + 1 HCLK cycles) for write accesses.
7-4	ADDHLD	Duration of the middle phase of the write access (ADDHLD HCLK cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 1.

Muxed mode - multiplexed asynchronous access to NOR Flash memory

Figure 21. Muxed read access waveforms

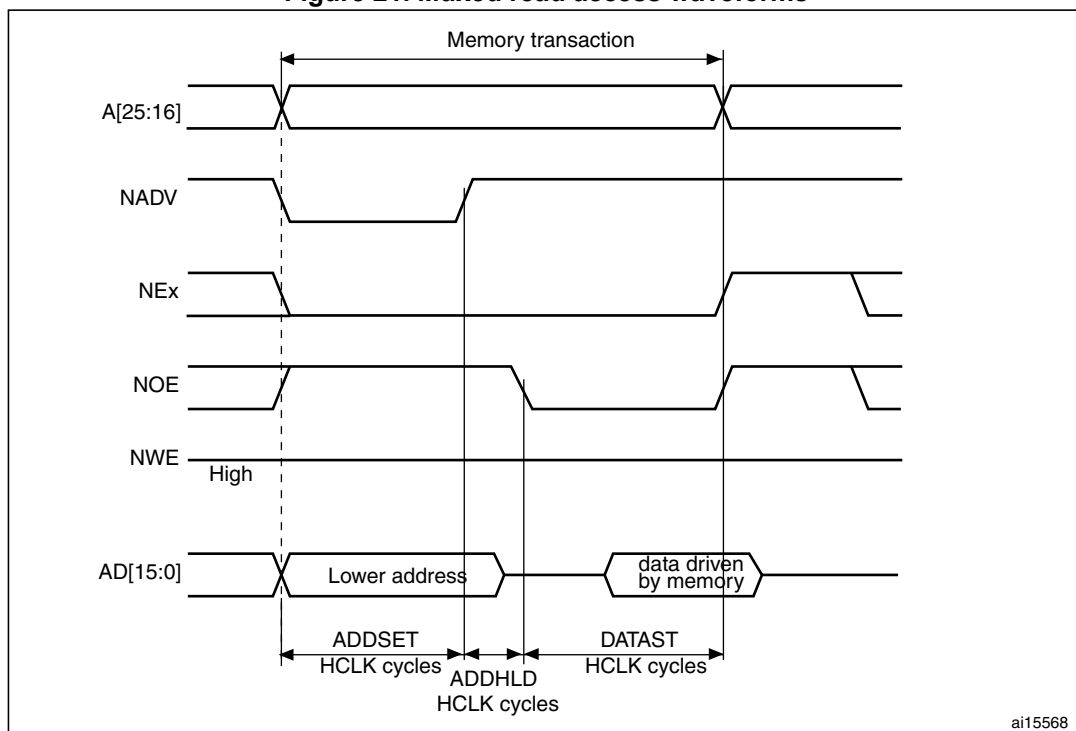
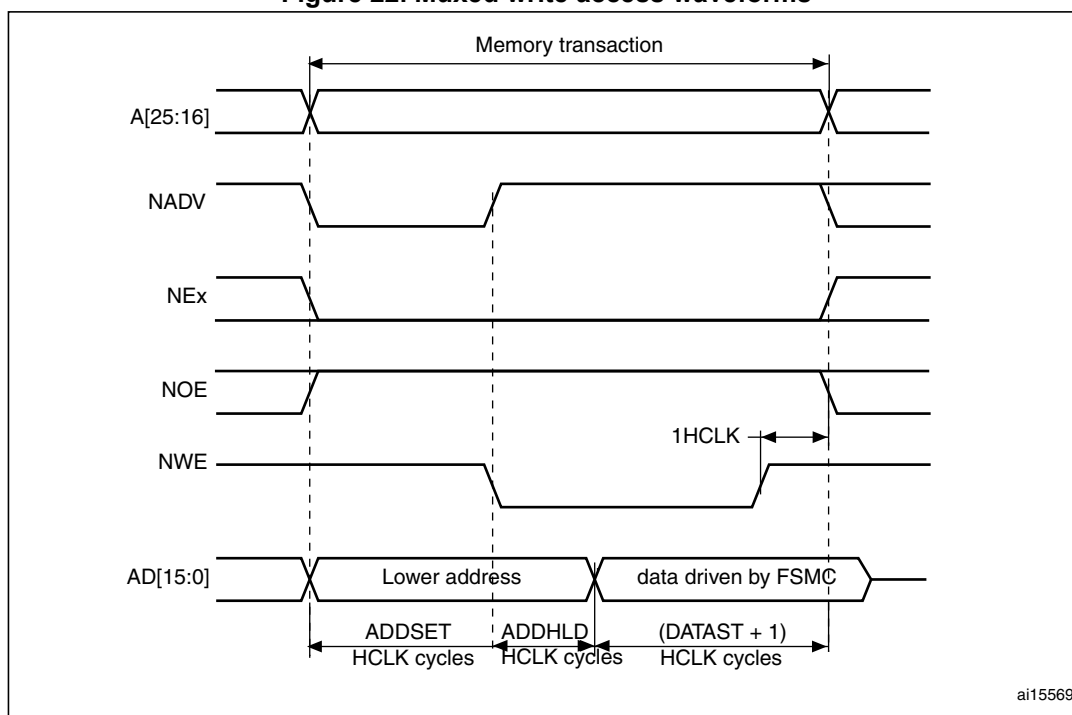


Figure 22. Muxed write access waveforms



The difference with mode D is the drive of the lower address byte(s) on the data bus.

Table 39. FMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-21	Reserved	0x000
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	Reserved	0x0
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	0x2 (NOR Flash memory)

Table 39. FMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
1	MUXEN	0x1
0	MBKEN	0x1

Table 40. FMC_BTRx bit fields

Bit No.	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles for read accesses and DATAST+1 HCLK cycles for write accesses).
7-4	ADDHLD	Duration of the middle phase of the access (ADDHLD HCLK cycles).
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 1.

WAIT management in asynchronous accesses

If the asynchronous memory asserts the WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase), programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data setup phase must be programmed so that WAIT can be detected 4 HCLK cycles before the end of the memory transaction. The following cases must be considered:

1. The memory asserts the WAIT signal aligned to NOE/NWE which toggles:

$$\text{DATAST} \geq (4 \times \text{HCLK}) + \text{max_wait_assertion_time}$$
2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):
 if

$$\text{max_wait_assertion_time} > \text{address_phase} + \text{hold_phase}$$

then:

$$\text{DATAST} \geq (4 \times \text{HCLK}) + (\text{max_wait_assertion_time} - \text{address_phase} - \text{hold_phase})$$

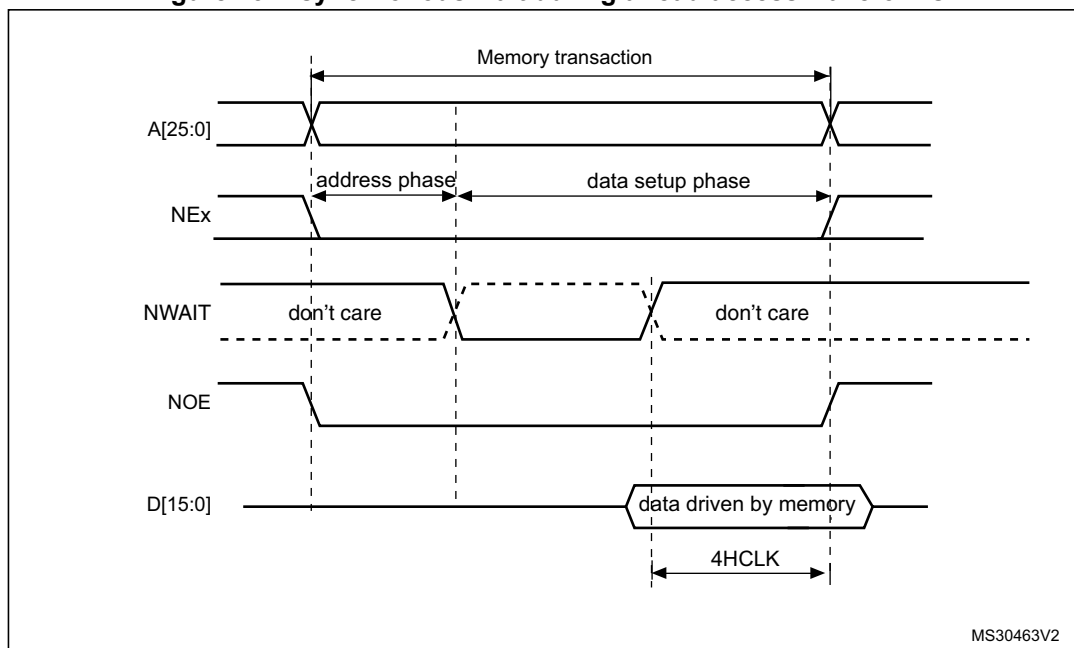
otherwise

$$\text{DATAST} \geq 4 \times \text{HCLK}$$

where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

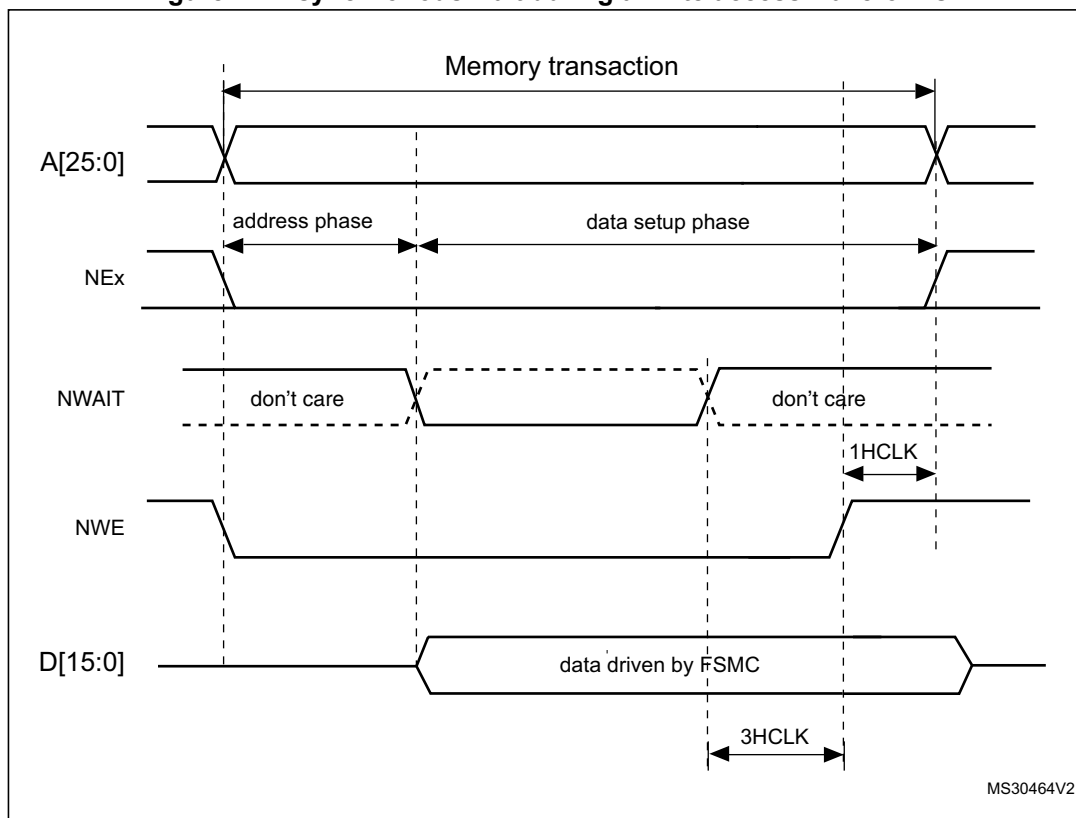
Figure 23 and Figure 24 show the number of HCLK clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

Figure 23. Asynchronous wait during a read access waveforms



1. N WAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

Figure 24. Asynchronous wait during a write access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

7.5.5 Synchronous transactions

The memory clock, FMC_CLK, is a submultiple of HCLK. It depends on the value of CLKDIV and the MWID/ AHB data size, following the formula given below:

$$\text{FMC_CLK divider ratio} = \max(\text{CLKDIV} + 1, \text{MWID}(\text{AHB data size}))$$

Whatever WID size: 16 or 8-bit, the FMC_CLK divider ratio is always defined by the programmed CLKDIV value.

Example:

- If CLKDIV=1, MWID = 16 bits, AHB data size=8 bits, FMC_CLK=HCLK/2.

NOR Flash memories specify a minimum time from NADV assertion to CLK high. To meet this constraint, the FMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

Data latency versus NOR memory latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR Flash configuration register. The FMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR Flash memories include the NADV Low cycle in the data latency count, so that the exact relation between the NOR Flash latency and the FMC DATLAT parameter can be either:

- NOR Flash latency = (DATLAT + 2) CLK clock cycles
- or NOR Flash latency = (DATLAT + 3) CLK clock cycles

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FMC samples the data and waits long enough to evaluate if the data are valid. Thus the FMC detects when the memory exits latency and real data are processed.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

Single-burst transfer

When the selected bank is configured in burst mode for synchronous accesses, if for example an AHB single-burst transaction is requested on 16-bit memories, the FMC performs a burst transaction of length 1 (if the AHB transfer is 16 bits), or length 2 (if the AHB transfer is 32 bits) and de-assert the Chip Select signal when the last data is strobed.

Such transfers are not the most efficient in terms of cycles compared to asynchronous read operations. Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

Wait management

For synchronous NOR Flash memories, NWAIT is evaluated after the programmed latency period, which corresponds to (DATLAT+2) CLK clock cycles.

If NWAIT is active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait states are inserted until NWAIT is inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the Chip Select and output enable signals valid. It does not consider the data as valid.

In burst mode, there are two timing configurations for the NOR Flash NWAIT signal:

- The Flash memory asserts the NWAIT signal one data cycle before the wait state (default after reset).
- The Flash memory asserts the NWAIT signal during the wait state

The FMC supports both NOR Flash wait state configurations, for each Chip Select, thanks to the WAITCFG bit in the FMC_BCRx registers (x = 0..3).

Figure 25. Wait configuration waveforms

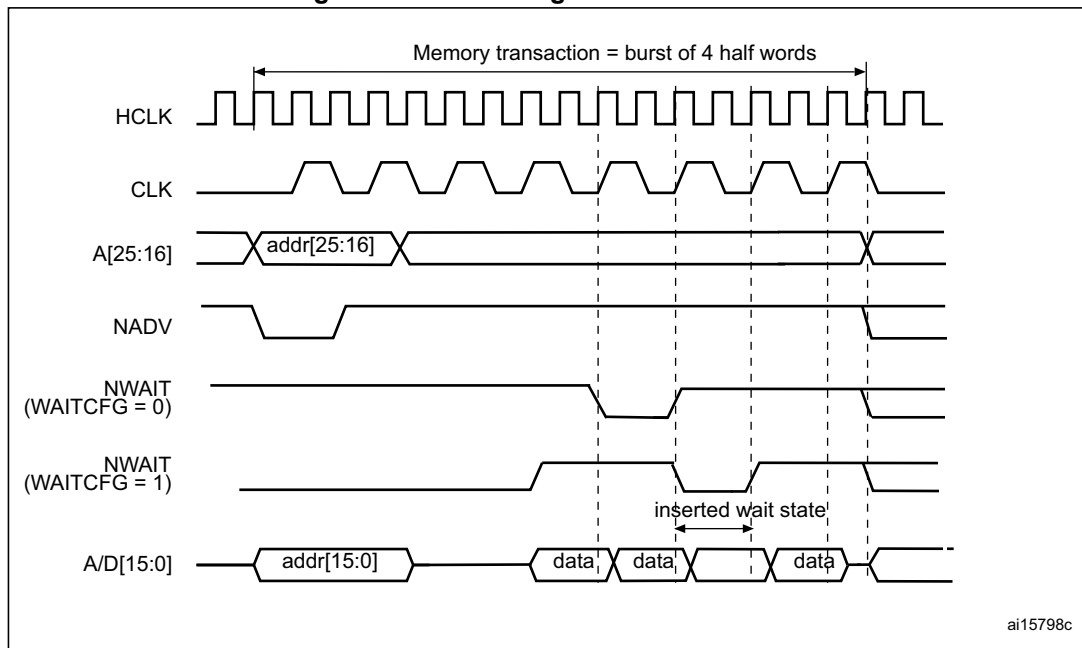
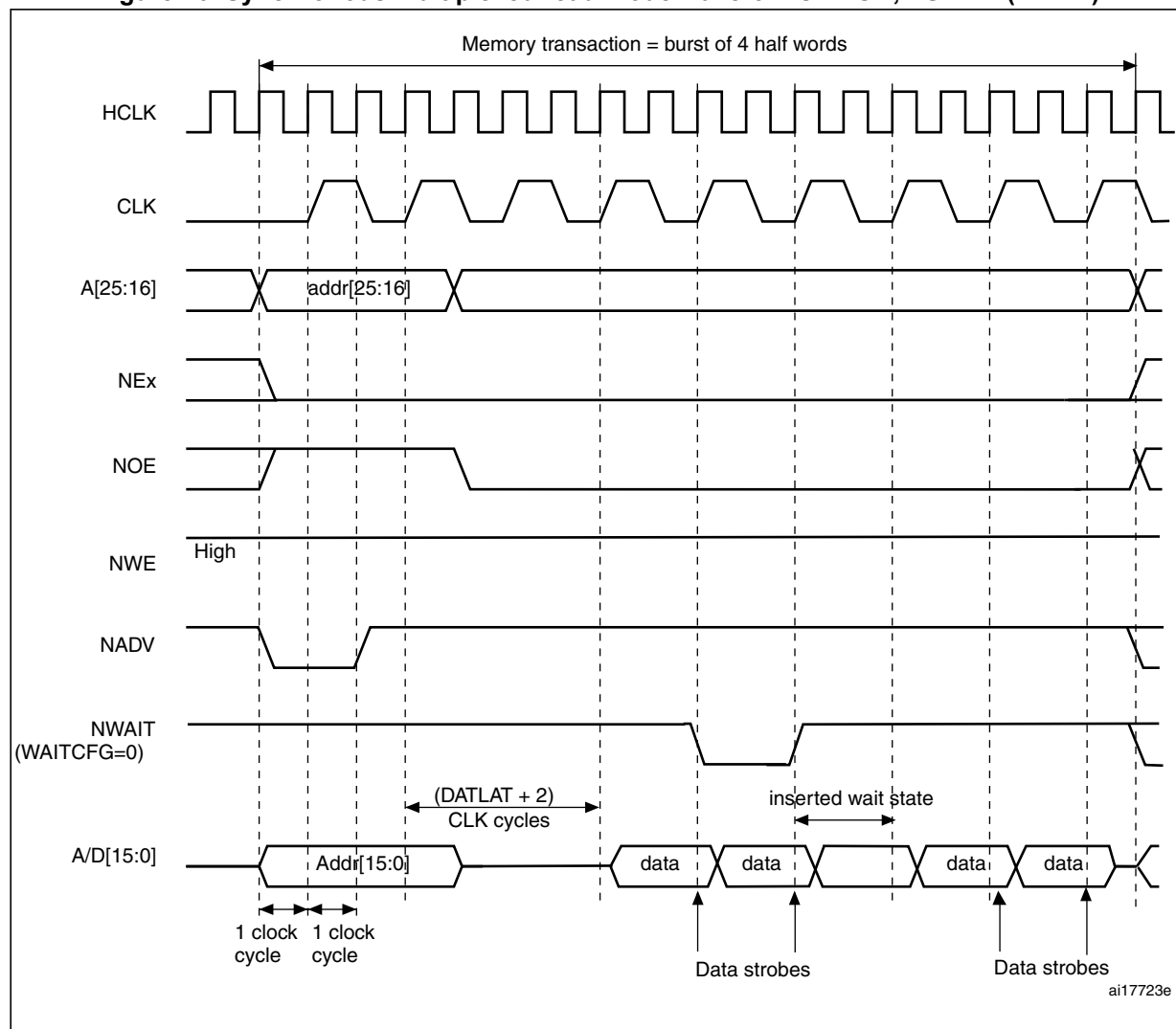


Figure 26. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)

1. Byte lane outputs (NBL are not shown; for NOR access, they are held high, and, for PSRAM (CRAM) access, they are held low.

Table 41. FMC_BCRx bit fields

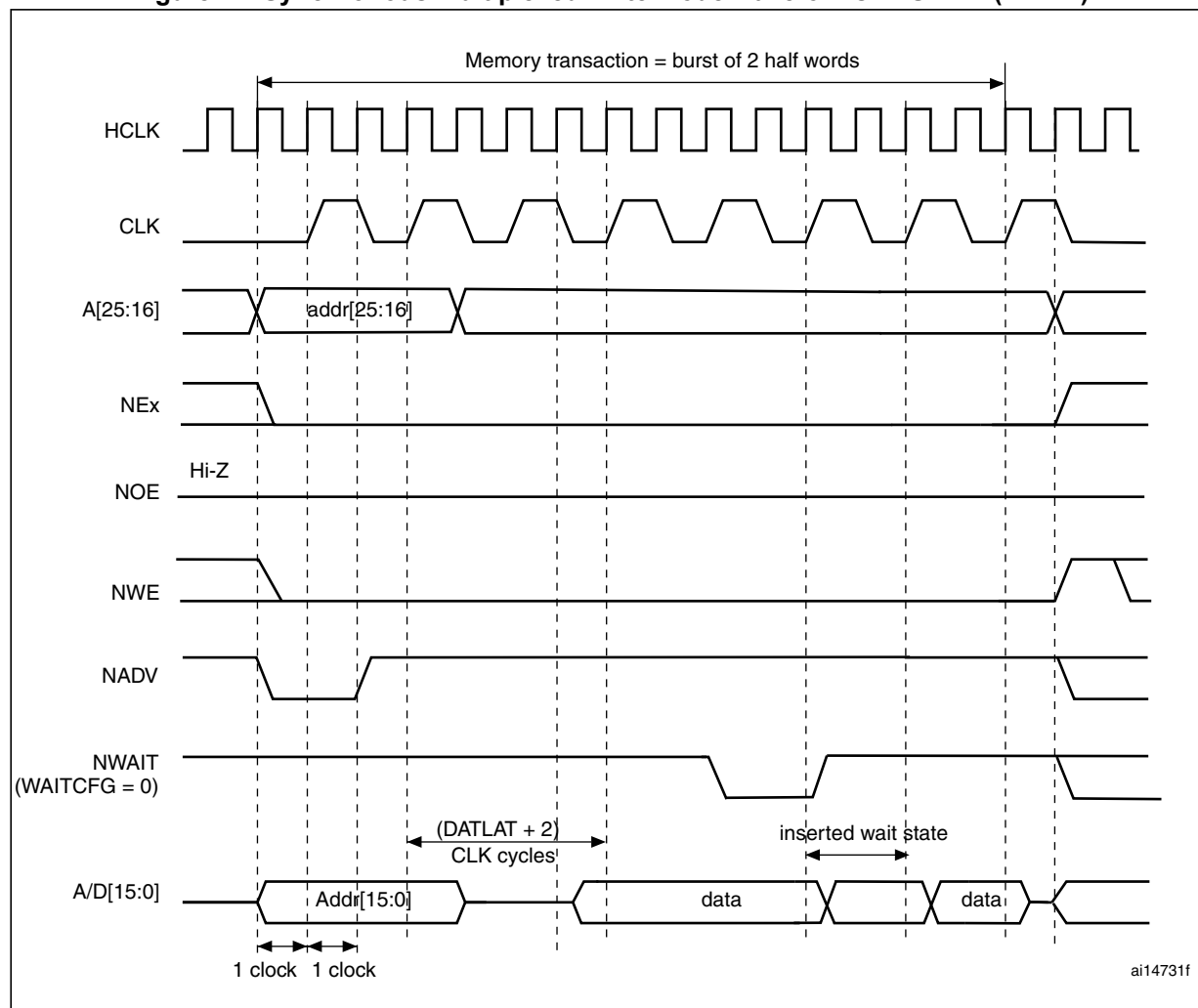
Bit No.	Bit name	Value to set
31-21	Reserved	0x000
20	CCLKEN	As needed
19	CBURSTRW	No effect on synchronous read
18-15	Reserved	0x0
14	EXTMOD	0x0
13	WAITEN	to be set to 1 if the memory supports this feature, to be kept at 0 otherwise
12	WREN	no effect on synchronous read
11	WAITCFG	to be set according to memory

Table 41. FMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
10	WRAPMOD	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1
6	FACCEN	Set according to memory support (NOR Flash memory)
5-4	MWID	As needed
3-2	MTYP	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 42. FMC_BTRx bit fields

Bit No.	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK (Not supported) 0x1 to get CLK = 2 × HCLK ..
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Don't care
7-4	ADDHLD	Don't care
3-0	ADDSET	Don't care

Figure 27. Synchronous multiplexed write mode waveforms - PSRAM (CRAM)

1. The memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.
2. Byte Lane (NBL) outputs are not shown, they are held low while NEx is active.

Table 43. FMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-20	Reserved	0x000
20	CCLKEN	As needed
19	CBURSTRW	0x1
18-15	Reserved	0x0
14	EXTMOD	0x0
13	WAITEN	to be set to 1 if the memory supports this feature, to be kept at 0 otherwise.
12	WREN	0x1
11	WAITCFG	0x0

Table 43. FMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
10	WRAPMOD	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	no effect on synchronous write
7	Reserved	0x1
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 44. FMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-30	Reserved	0x0
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK (not supported) 0x1 to get CLK = 2 × HCLK
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Don't care
7-4	ADDHLD	Don't care
3-0	ADDSET	Don't care

7.5.6 NOR/PSRAM controller registers

SRAM/NOR-Flash chip-select control registers 1..4 (FMC_BCR1..4)

Address offset: $8 * (x - 1)$, $x = 1...4$

Reset value: 0x0000 30DB for Bank1 and 0x0000 30D2 for Bank 2 to 4

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR Flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
Reserved											CCLKEN	CBURSTRW	Reserved				ASYCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN										
											rw	rw					rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
											rw	rw					rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31: 21 Reserved, must be kept at reset value

Bit 20 **CCLKEN**: Continuous Clock Enable.

This bit enables the FMC_CLK clock output to external memory devices.

0: The FMC_CLK is only generated during the synchronous memory access (read/write transaction). The FMC_CLK clock ratio is specified by the programmed CLKDIV value in the FMC_BCRx register (default after reset) .

1: The FMC_CLK is generated continuously during asynchronous and synchronous access. The FMC_CLK clock is activated when the CCLKEN is set.

Note: The CCLKEN bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register. Bank 1 must be configured in synchronous mode to generate the FMC_CLK continuous clock.

Note: If CCLKEN bit is set, the FMC_CLK clock ratio is specified by CLKDIV value in the FMC_BTR1 register. CLKDIV in FMC_BWTR1 is don't care.

Note: If the synchronous mode is used and CCLKEN bit is set, the synchronous memories connected to other banks than Bank 1 are clocked by the same clock (the CLKDIV value in the FMC_BTR2..4 and FMC_BWTR2..4 registers for other banks has no effect.)

Bit 19 **CBURSTRW**: Write burst enable.

For PSRAM (CRAM) operating in burst mode, the bit enables synchronous accesses during write operations. The enable bit for synchronous read accesses is the BURSTEN bit in the FMC_BCRx register.

0: Write operations are always performed in asynchronous mode

1: Write operations are performed in synchronous mode.

Bits 18:16 Reserved, must be kept at reset value

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers

This bit enables/disables the FMC to use the wait signal even during an asynchronous protocol.

0: NWAIT signal is not taken in to account when running an asynchronous protocol (default after reset)

1: NWAIT signal is taken in to account when running an asynchronous protocol

Bit 14 EXTMOD: Extended mode enable.

This bit enables the FMC to program the write timings for asynchronous accesses inside the FMC_BWTR register, thus resulting in different timings for read and write operations.

- 0: values inside FMC_BWTR register are not taken into account (default after reset)
- 1: values inside FMC_BWTR register are taken into account

Note: When the extended mode is disabled, the FMC can operate in Mode1 or Mode2 as follows:

- *Mode 1 is the default mode when the SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01)*
- *Mode 2 is the default mode when the NOR memory type is selected (MTYP = 0x10).*

Bit 13 WAITEN: Wait enable bit.

This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in synchronous mode.

- 0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed Flash latency period)
- 1: NWAIT signal is enabled (its level is taken into account after the programmed latency period to insert wait states if asserted) (default after reset)

Bit 12 WREN: Write enable bit.

This bit indicates whether write operations are enabled/disabled in the bank by the FMC:

- 0: Write operations are disabled in the bank by the FMC, an AHB error is reported,
- 1: Write operations are enabled for the bank by the FMC (default after reset).

Bit 11 WAITCFG: Wait timing configuration.

The NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted when accessing the memory in synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:

- 0: NWAIT signal is active one data cycle before wait state (default after reset),
- 1: NWAIT signal is active during wait state (not used for PSRAM).

Bit 10 WRAPMOD: Wrapped burst mode support.

Defines whether the controller will or not split an AHB burst wrap access into two linear accesses. Valid only when accessing memories in burst mode

- 0: Direct wrapped burst is not enabled (default after reset),
- 1: Direct wrapped burst is enabled.

Note: This bit has no effect as the CPU and DMA cannot generate wrapping burst transfers.

Bit 9 WAITPOL: Wait signal polarity bit.

Defines the polarity of the wait signal from memory used for either in synchronous or asynchronous mode:

- 0: NWAIT active low (default after reset),
- 1: NWAIT active high.

Bit 8 BURSTEN: Burst enable bit.

This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in burst mode:

- 0: Burst mode disabled (default after reset). Read accesses are performed in asynchronous mode.
- 1: Burst mode enable. Read accesses are performed in synchronous mode.

Bit 7 Reserved, must be kept at reset value**Bit 6 FACCEN:** Flash access enable

Enables NOR Flash memory access operations.

- 0: Corresponding NOR Flash memory access is disabled
- 1: Corresponding NOR Flash memory access is enabled (default after reset)

Bits 5:4 **MWID**: Memory data bus width.

Defines the external memory device width, valid for all type of memories.

- 00: 8 bits
- 01: 16 bits (default after reset)
- 10: reserved, do not use
- 11: reserved, do not use

Bits 3:2 **MTYP**: Memory type.

Defines the type of external memory attached to the corresponding memory bank:

- 00: SRAM (default after reset for Bank 2...4)
- 01: PSRAM (CRAM)
- 10: NOR Flash/OneNAND Flash (default after reset for Bank 1)
- 11: reserved

Bit 1 **MUXEN**: Address/data multiplexing enable bit.

When this bit is set, the address and data values are multiplexed on the data bus, valid only with NOR and PSRAM memories:

- 0: Address/Data nonmultiplexed
- 1: Address/Data multiplexed on databus (default after reset)

Bit 0 **MBKEN**: Memory bank enable bit.

Enables the memory bank. After reset Bank1 is enabled, all others are disabled. Accessing a disabled bank causes an ERROR on AHB bus.

- 0: Corresponding memory bank is disabled
- 1: Corresponding memory bank is enabled

SRAM/NOR-Flash chip-select timing registers 1..4 (FMC_BTR1..4)

Address offset: $0x04 + 8 * (x - 1)$, $x = 1..4$

Reset value: 0x0FFF FFFF

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR Flash memories. If the EXTMOD bit is set in the FMC_BCRx register, then this register is partitioned for write and read access, that is, 2 registers are available: one to configure read accesses (this register) and one to configure write accesses (FMC_BWTRx registers).

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

Bits 31:30 Reserved, must be kept at reset value

Bits 29:28 **ACCMOD**: Access mode

Specifies the asynchronous access modes as shown in the timing diagrams. These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

- 00: access mode A
- 01: access mode B
- 10: access mode C
- 11: access mode D

Bits 27:24 **DATLAT**: (see note below bit descriptions): Data latency for synchronous memory

For synchronous access with read/write burst mode enabled (BURSTEN / CBURSTRW bits set), defines the number of memory clock cycles (+2) to issue to the memory before reading/writing the first data:

This timing parameter is not expressed in HCLK periods, but in FMC_CLK periods.

For asynchronous access, this value is don't care.

- 0000: Data latency of 2 CLK clock cycles for first burst access
- 1111: Data latency of 17 CLK clock cycles for first burst access (default value after reset)

Bits 23:20 **CLKDIV**: Clock divide ratio (for FMC_CLK signal)

Defines the period of FMC_CLK clock output signal, expressed in number of HCLK cycles:

- 0000: Reserved
- 0001: FMC_CLK period = 2 × HCLK periods
- 0010: FMC_CLK period = 3 × HCLK periods
- 1111: FMC_CLK period = 16 × HCLK periods (default value after reset)

In asynchronous NOR Flash, SRAM or PSRAM accesses, this value is don't care.

Note: Refer to [Section 7.5.5: Synchronous transactions for FMC_CLK divider ratio formula](#)

Bits 19:16 **BUSTURN**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write/read transaction. This delay allows to match the minimum time between consecutive transactions (t_{EHEL} from NEx high to NEx low) and the maximum time needed by the memory to free the data bus after a read access (t_{EHQZ}):

$(BUSTURN + 1)HCLK \text{ period} \geq t_{EHELmin}$ and $(BUSTURN + 2)HCLK \text{ period} \geq t_{EHQZmax}$ if EXTMOD = '0'

$(BUSTURN + 2)HCLK \text{ period} \geq \max(t_{EHELmin}, t_{EHQZmax})$ if EXTMOD = '1'.

0000: BUSTURN phase duration = 0 HCLK clock cycle added

...

1111: BUSTURN phase duration = 15 × HCLK clock cycles (default value after reset)

Bits 15:8 DATAST: Data-phase duration

These bits are written by software to define the duration of the data phase (refer to [Figure 10](#) to [Figure 22](#)), used in asynchronous accesses:

0000 0000: Reserved

0000 0001: DATAST phase duration = 1 × HCLK clock cycles

0000 0010: DATAST phase duration = 2 × HCLK clock cycles

...

1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

For each memory type and access mode data-phase duration, please refer to the respective figure ([Figure 10](#) to [Figure 22](#)).

Example: Mode1, write access, DATAST=1: Data-phase duration= DATAST+1 = 2 HCLK clock cycles.

Note: In synchronous accesses, this value is don't care.

Bits 7:4 ADDHLD: Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to [Figure 10](#) to [Figure 22](#)), used in mode D or multiplexed accesses:

0000: Reserved

0001: ADDHLD phase duration = 1 × HCLK clock cycle

0010: ADDHLD phase duration = 2 × HCLK clock cycle

...

1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

For each access mode address-hold phase duration, please refer to the respective figure ([Figure 10](#) to [Figure 22](#)).

Note: In synchronous accesses, this value is not used, the address hold phase is always 1 memory clock period duration.

Bits 3:0 ADDSET: Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to [Figure 10](#) to [Figure 22](#)), used in SRAMs, ROMs and asynchronous NOR Flash:

0000: ADDSET phase duration = 0 × HCLK clock cycle

...

1111: ADDSET phase duration = 15 × HCLK clock cycles (default value after reset)

For each access mode address setup phase duration, please refer to the respective figure (refer to [Figure 10](#) to [Figure 22](#)).

Note: In synchronous accesses, this value is don't care.

In Muxed mode or Mode D, the minimum value for ADDSET is 1.

Note: *PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to prolong the latency as needed.*

With PSRAMs (CRAMs) the filled DATLAT must be set to 0, so that the FMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write when the memory is ready.

This method can be used also with the latest generation of synchronous Flash memories that issue the NWAIT signal, unlike older Flash memories (check the datasheet of the specific Flash memory being used).

SRAM/NOR-Flash write timing registers 1..4 (FMC_BWTR1..4)

Address offset: $0x104 + 8 * (x - 1)$, $x = 1...4$

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank. It is used for SRAMs, PSRAMs and NOR Flash memories. When the EXTMOD bit is set in the FMC_BCRx register, then this register is active for write access.

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

Bits 31:30 Reserved, must be kept at reset value

Bits 29:28 **ACCMOD**: Access mode.

Specifies the asynchronous access modes as shown in the next timing diagrams. These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: access mode A

01: access mode B

10: access mode C

11: access mode D

Bits 27:20 Reserved, must be kept at reset value

Bits 19:16 **BUSTURN**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write transaction to match the minimum time between consecutive transactions (t_{EHEL} from ENx high to ENx low):

(BUSTURN + 1) HCLK period $\geq t_{EHELmin}$

0000: BUSTURN phase duration = 0 HCLK clock cycle added

...

1111: BUSTURN phase duration = 15 HCLK clock cycles added (default value after reset)

Bits 15:8 **DATAST**: Data-phase duration.

These bits are written by software to define the duration of the data phase (refer to [Figure 10](#) to [Figure 22](#)), used in asynchronous SRAM, PSRAM and NOR Flash memory accesses:

0000 0000: Reserved

0000 0001: DATAST phase duration = 1 × HCLK clock cycles

0000 0010: DATAST phase duration = 2 × HCLK clock cycles

...

1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

Bits 7:4 **ADDHLD**: Address-hold phase duration.

These bits are written by software to define the duration of the *address hold* phase (refer to [Figure 10](#) to [Figure 22](#)), used in asynchronous multiplexed accesses:

0000: Reserved

0001: ADDHLD phase duration = 1 × HCLK clock cycle

0010: ADDHLD phase duration = 2 × HCLK clock cycle

...

1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

Note: In synchronous NOR Flash accesses, this value is not used, the address hold phase is always 1 Flash clock period duration.

Bits 3:0 **ADDSET**: Address setup phase duration.

These bits are written by software to define the duration of the *address setup* phase in HCLK cycles (refer to [Figure 10](#) to [Figure 22](#)), used in asynchronous accesses:

0000: ADDSET phase duration = $0 \times$ HCLK clock cycle

...

1111: ADDSET phase duration = $15 \times$ HCLK clock cycles (default value after reset)

Note: In synchronous accesses, this value is not used, the address setup phase is always 1 Flash clock period duration. In muxed mode, the minimum ADDSET value is 1.

7.6 NAND Flash/PC Card controller

The FMC generates the appropriate signal timings to drive the following types of device:

- 8- and 16-bit NAND Flash memories
- 16-bit PC Card compatible devices

The NAND Flash/PC Card controller can control three external banks, Bank 2, 3 and 4:

- Bank 2 and Bank 3 support NAND Flash devices
- Bank 4 supports PC Card devices.

Each bank is configured through dedicated registers ([Section 7.6.8](#)). The programmable memory parameters include access timings (shown in [Table 45](#)) and ECC configuration.

Table 45. Programmable NAND Flash/PC Card access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Memory setup time	Number of clock cycles (HCLK) required to set up the address before the command assertion	Read/Write	AHB clock cycle (HCLK)	1	256
Memory wait	Minimum duration (in HCLK clock cycles) of the command assertion	Read/Write	AHB clock cycle (HCLK)	2	256
Memory hold	Number of clock cycles (HCLK) during which the address must be held (as well as the data if a write access is performed) after the command de-assertion	Read/Write	AHB clock cycle (HCLK)	1	255
Memory databus high-Z	Number of clock cycles (HCLK) during which the data bus is kept in high-Z state after a write access has started	Write	AHB clock cycle (HCLK)	0	255

7.6.1 External memory interface signals

The following tables list the signals that are typically used to interface NAND Flash memory and PC Card.

Note: The prefix "N" identifies the signals which are active low.

8-bit NAND Flash memory

Table 46. 8-bit NAND Flash

FMC signal name	I/O	Function
A[17]	O	NAND Flash address latch enable (ALE) signal
A[16]	O	NAND Flash command latch enable (CLE) signal
D[7:0]	I/O	8-bit multiplexed, bidirectional address/data bus
NCE[x]	O	Chip Select, x = 2, 3
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT[3:2]	I	NAND Flash ready/busy input signal to the FMC

Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

16-bit NAND Flash memory**Table 47. 16-bit NAND Flash**

FMC signal name	I/O	Function
A[17]	O	NAND Flash address latch enable (ALE) signal
A[16]	O	NAND Flash command latch enable (CLE) signal
D[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NCE[x]	O	Chip Select, x = 2, 3
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT[3:2]	I	NAND Flash ready/busy input signal to the FMC

Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

Table 48. 16-bit PC Card

FMC signal name	I/O	Function
A[10:0]	O	Address bus
NIORD	O	Output enable for I/O space
NIOWR	O	Write enable for I/O space
NREG	O	Register signal indicating if access is in Common or Attribute space
D[15:0]	I/O	Bidirectional databus
NCE4_1	O	Chip Select 1
NCE4_2	O	Chip Select 2 (indicates if access is 16-bit or 8-bit)
NOE	O	Output enable in Common and in Attribute space
NWE	O	Write enable in Common and in Attribute space
NWAIT	I	PC Card wait input signal to the FMC (memory signal name IORDY)
INTR	I	PC Card interrupt to the FMC (only for PC Cards that can generate an interrupt)
CD	I	PC Card presence detection. Active high. If an access is performed to the PC Card banks while CD is low, an AHB error is generated. Refer to Section 7.3: AHB interface

7.6.2 NAND Flash / PC Card supported memories and transactions

Table 49 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND Flash / PC Card controller are shown in gray.

Table 49. Supported memories and transactions

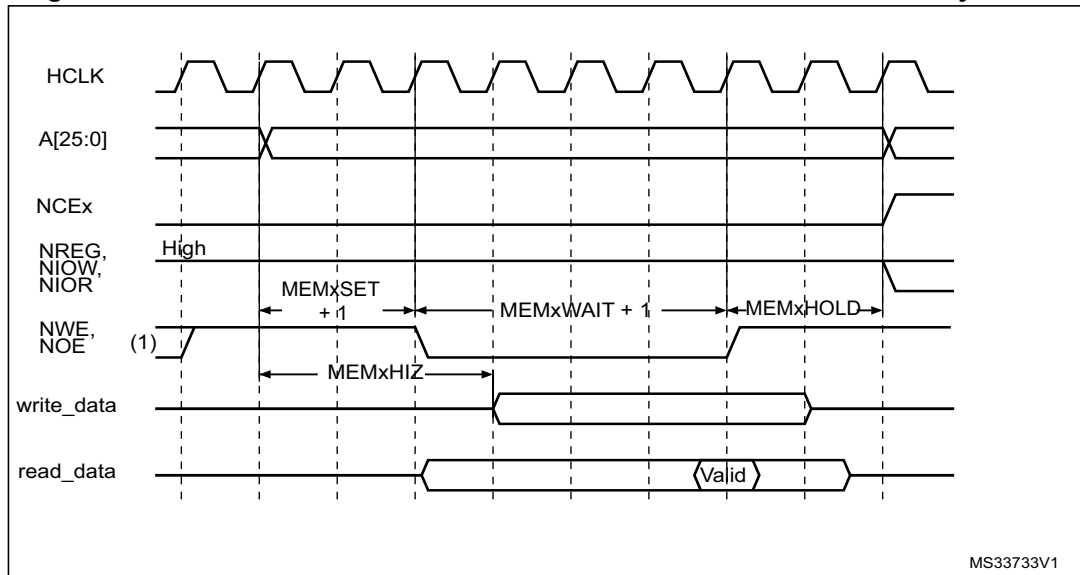
Device	Mode	R/W	AHB data size	Memory data size	Allowed/not allowed	Comments
NAND 8-bit	Asynchronous	R	8	8	Y	
	Asynchronous	W	8	8	Y	
	Asynchronous	R	16	8	Y	Split into 2 FMC accesses
	Asynchronous	W	16	8	Y	Split into 2 FMC accesses
	Asynchronous	R	32	8	Y	Split into 4 FMC accesses
	Asynchronous	W	32	8	Y	Split into 4 FMC accesses
NAND 16-bit	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	N	
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses

7.6.3 Timing diagrams for NAND Flash memory and PC Card

Each PC Card/CompactFlash and NAND Flash memory bank is managed through a set of registers:

- Control register: FMC_PCRx
- Interrupt status register: FMC_SRx
- ECC register: FMC_ECCRx
- Timing register for Common memory space: FMC_PMEMx
- Timing register for Attribute memory space: FMC_PATTx
- Timing register for I/O space: FMC_PIOx

Each timing configuration register contains three parameters used to define number of HCLK cycles for the three phases of any PC Card/CompactFlash or NAND Flash access, plus one parameter that defines the timing for starting driving the data bus when a write access is performed. [Figure 28](#) shows the timing parameter definitions for common memory accesses, knowing that Attribute and I/O (only for PC Card) memory space access timings are similar.

Figure 28. NAND Flash/PC Card controller waveforms for common memory access

1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.

7.6.4 NAND Flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND Flash memory device are driven by address signals from the FMC controller. This means that to send a command or an address to the NAND Flash memory, the CPU has to perform a write to a specific address in its memory space.

A typical page read operation from the NAND Flash device requires the following steps:

3. Program and enable the corresponding memory bank by configuring the FMC_PCRx and FMC_PMEMx (and for some devices, FMC_PATTx, see [Section 7.6.5: NAND Flash prewait functionality](#)) registers according to the characteristics of the NAND Flash memory (PWID bits for the data bus width of the NAND Flash, PTYP = 1, PWAITEN = 0 or 1 as needed, see [Section 7.4.2: NAND Flash memory/PC Card address mapping](#) for timing configuration).
4. The CPU performs a byte write to the common memory space, with data byte equal to one Flash command byte (for example 0x00 for Samsung NAND Flash devices). The LE input of the NAND Flash memory is active during the write strobe (low pulse on NWE), thus the written byte is interpreted as a command by the NAND Flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.
5. The CPU can send the start address (STARTAD) for a read operation by writing four bytes (or three for smaller capacity devices), STARTAD[7:0], STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] (for 64 Mb x 8 bit NAND Flash memories) in the common memory or attribute space. The ALE input of the NAND Flash device is active during the write strobe (low pulse on NWE), thus the written bytes are interpreted as the start address for read operations. Using the attribute memory space makes it possible to use a different timing configuration of the FMC, which can be used

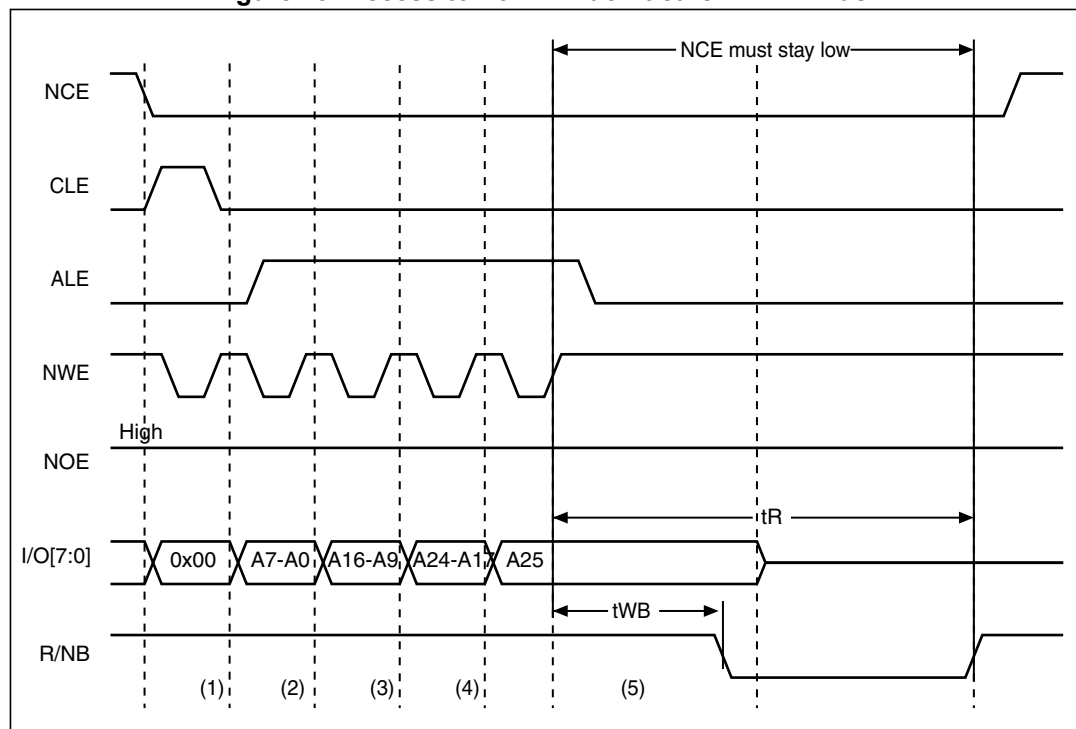
to implement the prewait functionality needed by some NAND Flash memories (see details in [Section 7.6.5: NAND Flash prewait functionality](#)).

6. The controller waits for the NAND Flash memory to be ready (R/NB signal high), before starting a new access to the same or another memory bank. While waiting, the controller holds the NCE signal active (low).
7. The CPU can then perform byte read operations from the common memory space to read the NAND Flash page (data field + Spare field) byte by byte.
8. The next NAND Flash page can be read without any CPU command or address write operation. This can be done in three different ways:
 - by simply performing the operation described in step 5
 - a new random address can be accessed by restarting the operation at step 3
 - a new command can be sent to the NAND Flash device by restarting at step 2

7.6.5 NAND Flash prewait functionality

Some NAND Flash devices require that, after writing the last part of the address, the controller waits for the R/NB signal to go low. (see [Figure 29](#)).

Figure 29. Access to non 'CE don't care' NAND-Flash



1. CPU wrote byte 0x00 at address 0x7001 0000.
2. CPU wrote byte A7~A0 at address 0x7002 0000.
3. CPU wrote byte A16~A9 at address 0x7002 0000.
4. CPU wrote byte A24~A17 at address 0x7002 0000.
5. CPU wrote byte A25 at address 0x7802 0000: FMC performs a write access using FMC_PATT2 timing definition, where $ATTHOLD \geq 7$ (providing that $(7+1) \times HCLK = 112 \text{ ns} > t_{WB} \text{ max}$). This guarantees that NCE remains low until R/NB goes low and high again (only requested for NAND Flash memories where NCE is not don't care).

When this functionality is required, it can be ensured by programming the MEMHOLD value to meet the t_{WB} timing. However any CPU read or write access to the NAND Flash memory has a hold delay of (MEMHOLD + 1) HCLK cycles inserted between the rising edge of the NWE signal and the next access.

To cope with this timing constraint, the attribute memory space can be used by programming its timing register with an ATTHOLD value that meets the t_{WB} timing, and by keeping the MEMHOLD value at its minimum value. The CPU must then use the common memory space for all NAND Flash read and write accesses, except when writing the last address byte to the NAND Flash device, where the CPU must write to the attribute memory space.

7.6.6 Computation of the error correction code (ECC) in NAND Flash memory

The FMC PC Card controller includes two error correction code computation hardware blocks, one per memory bank. They reduce the host CPU workload when processing the ECC by software.

These two ECC blocks are identical and associated with Bank 2 and Bank 3. As a consequence, no hardware ECC computation is available for memories connected to Bank 4.

The ECC algorithm implemented in the FMC can perform 1-bit error correction and 2-bit error detection per 256, 512, 1 024, 2 048, 4 096 or 8 192 bytes read or written from/to the NAND Flash memory. It is based on the Hamming coding algorithm and consists in calculating the row and column parity.

The ECC modules monitor the NAND Flash data bus and read/write signals (NCE and NWE) each time the NAND Flash memory bank is active.

The ECC operates as follows:

- When accessing NAND Flash memory bank 2 or bank 3, the data present on the D[15:0] bus is latched and used for ECC computation.
- When accessing any other address in NAND Flash memory, the ECC logic is idle, and does not perform any operation. As a result, write operations to define commands or addresses to the NAND Flash memory are not taken into account for ECC computation.

Once the desired number of bytes has been read/written from/to the NAND Flash memory by the host CPU, the FMC_ECCR2/3 registers must be read to retrieve the computed value. Once read, they should be cleared by resetting the ECCEN bit to '0'. To compute a new data block, the ECCEN bit must be set to one in the FMC_PCR2/3 registers.

To perform an ECC computation:

1. Enable the ECCEN bit in the FMC_PCR2/3 register.
2. Write data to the NAND Flash memory page. While the NAND page is written, the ECC block computes the ECC value.
3. Read the ECC value available in the FMC_ECCR2/3 register and store it in a variable.
4. Clear the ECCEN bit and then enable it in the FMC_PCR2/3 register before reading back the written data from the NAND page. While the NAND page is read, the ECC block computes the ECC value.
5. Read the new ECC value available in the FMC_ECCR2/3 register.
6. If the two ECC values are the same, no correction is required, otherwise there is an ECC error and the software correction routine returns information on whether the error can be corrected or not.

7.6.7 PC Card/CompactFlash operations

Address spaces and memory accesses

The FMC supports CompactFlash devices and PC Cards in Memory mode and I/O mode (True IDE mode is not supported).

The CompactFlash and PC Cards are made of 3 memory spaces:

- Common Memory space
- Attribute space
- I/O Memory space

The nCE2 and nCE1 pins (FMC_NCE4_2 and FMC_NCE4_1 respectively) select the card and indicate whether a byte or a word operation is being performed: nCE2 accesses the odd byte on D15-8 and nCE1 accesses the even byte on D7-0 if A0=0 or the odd byte on D7-0 if A0=1. The full word is accessed on D15-0 if both nCE2 and nCE1 are low.

The memory space is selected by asserting low nOE for read accesses or nWE for write accesses, combined with the low assertion of nCE2/nCE1 and nREG.

- If pin nREG=1 during the memory access, the common memory space is selected
- If pin nREG=0 during the memory access, the attribute memory space is selected

The I/O space is selected by asserting nIORD space for read accesses or nIOWR for write accesses [instead of nOE/nWE for memory space], combined with nCE2/nCE1. Note that nREG must also be asserted low when accessing I/O space.

Three type of accesses are allowed for a 16-bit PC Card:

- Accesses to Common Memory space for data storage can be either 8-bit accesses at even addresses or 16-bit AHB accesses.
Note that 8-bit accesses at odd addresses are not supported and nCE2 will not be driven low. A 32-bit AHB request is translated into two 16-bit memory accesses.
- Accesses to Attribute Memory space where the PC Card stores configuration information are limited to 8-bit AHB accesses at even addresses.

Note that a 16-bit AHB access will be converted into a single 8-bit memory transfer: nCE1 will be asserted low, nCE2 will be asserted high and only the even byte on D7-D0 will be valid. Instead a 32-bit AHB access will be converted into two 8-bit memory

transfers at even addresses: nCE1 will be asserted low, NCE2 will be asserted high and only the even bytes will be valid.

- Accesses to I/O space can be either 8-bit or 16 bit AHB accesses.

Table 50. 16-bit PC-Card signals and access type

nCE2	nCE1	nREG	nOE/nWE	nIORD /nIOWR	A10	A9	A7-1	A0	Space	Access type	Allowed/not Allowed
1	0	1	0	1	X	X	X-X	X	Common Memory Space	Read/Write byte on D7-D0	YES
0	1	1	0	1	X	X	X-X	X		Read/Write byte on D15-D8	Not supported
0	0	1	0	1	X	X	X-X	0		Read/Write word on D15-D0	YES
X	0	0	0	1	0	1	X-X	0	Attribute Space	Read or Write Configuration Registers	YES
X	0	0	0	1	0	0	X-X	0		Read or Write CIS (Card Information Structure)	YES
1	0	0	0	1	X	X	X-X	1	Attribute Space	Invalid Read or Write (odd address)	YES
0	1	0	0	1	X	X	X-X	x		Invalid Read or Write (odd address)	YES
1	0	0	1	0	X	X	X-X	0	I/O space	Read Even Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	1		Read Odd Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	0		Write Even Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	1		Write Odd Byte on D7-0	YES
0	0	0	1	0	X	X	X-X	0		Read Word on D15-0	YES
0	0	0	1	0	X	X	X-X	0		Write word on D15-0	YES
0	1	0	1	0	X	X	X-X	X		Read Odd Byte on D15-8	Not supported
0	1	0	1	0	X	X	X-X	X		Write Odd Byte on D15-8	Not supported

FMC Bank 4 gives access to those 3 memory spaces as described in [Section 7.4.2: NAND Flash memory/PC Card address mapping](#) and [Table 17: NAND/PC Card memory mapping and timing registers](#).

Wait feature

The CompactFlash or PC Card may request the FMC to extend the length of the access phase programmed by MEMWAITx/ATTWAITx/IOWAITx bits, asserting the nWAIT signal after nOE/nWE or nIORD/nIOWR activation if the wait feature is enabled through the

PWAITEN bit in the FMC_PCRx register. To detect correctly the nWAIT assertion, the MEMWAITx/ATTWAITx/IOWAITx bits must be programmed as follows:

$$xxWAITx \geq 4 + \frac{\text{max_wait_assertion_time}}{\text{HCLK}}$$

where max_wait_assertion_time is the maximum time taken by NWAIT to go low once nOE/nWE or nIORD/nIOWR is low.

After WAIT de-assertion, the FMC extends the WAIT phase for 4 HCLK clock cycles.

7.6.8 NAND Flash/PC Card controller registers

PC Card/NAND Flash control registers 2..4 (FMC_PCR2..4)

Address offset: 0x40 + 0x20 * (x - 1), x = 2..4

Reset value: 0x0000 0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												ECCPS			TAR				TCLR				Res.	ECCEN	PWID		PTYP	PBKEN	PWAITEN	Reserved	
												r/w	r/w	r/w	r/w	r/w	r/w	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 19:17 **ECCPS**: ECC page size.

Defines the page size for the extended ECC:

000: 256 bytes
 001: 512 bytes
 010: 1024 bytes
 011: 2048 bytes
 100: 4096 bytes
 101: 8192 bytes

Bits 16:13 **TAR**: ALE to RE delay.

Sets time from ALE low to RE low in number of AHB clock cycles (HCLK).

Time is: $t_{ar} = (TAR + SET + 2) \times THCLK$ where THCLK is the HCLK clock period

0000: 1 HCLK cycle (default)
 1111: 16 HCLK cycles

Note: SET is MEMSET or ATTSET according to the addressed space.

Bits 12:9 **TCLR**: CLE to RE delay.

Sets time from CLE low to RE low in number of AHB clock cycles (HCLK).

Time is: $t_{clr} = (TCLR + SET + 2) \times THCLK$ where THCLK is the HCLK clock period

0000: 1 HCLK cycle (default)
 1111: 16 HCLK cycles

Note: SET is MEMSET or ATTSET according to the addressed space.

Bits 8:7 Reserved, must be kept at reset value

Bit 6 **ECCEN**: ECC computation logic enable bit

0: ECC logic is disabled and reset (default after reset),
 1: ECC logic is enabled.

Bits 5:4 **PWID**: Data bus width.

Defines the external memory device width.

00: 8 bits

01: 16 bits (default after reset). This value is mandatory for PC Cards.

10: reserved, do not use

11: reserved, do not use

Bit 3 **PTYP**: Memory type.

Defines the type of device attached to the corresponding memory bank:

0: PC Card, CompactFlash, CF+ or PCMCIA

1: NAND Flash (default after reset)

Bit 2 **PBKEN**: PC Card/NAND Flash memory bank enable bit.

Enables the memory bank. Accessing a disabled memory bank causes an ERROR on AHB bus

0: Corresponding memory bank is disabled (default after reset)

1: Corresponding memory bank is enabled

Bit 1 **PWAITEN**: Wait feature enable bit.

Enables the Wait feature for the PC Card/NAND Flash memory bank:

0: disabled

1: enabled

Note: For a PC Card, when the wait feature is enabled, the MEMWAITx/ATTWAITx/IOWAITx bits must be programmed to a value as follows:

$xxWAITx \geq 4 + \max_wait_assertion_time/HCLK$

Where max_wait_assertion_time is the maximum time taken by NWAIT to go low once nOE/nWE or nLORD/nLOWR is low.

Bit 0 Reserved.

FIFO status and interrupt register 2..4 (FMC_SR2..4)

Address offset: $0x44 + 0x20 * (x-1)$, $x = 2..4$

Reset value: 0x0000 0040

This register contains information about the FIFO status and interrupt. The FMC features a FIFO that is used when writing to memories to transfer up to 16 words of data from the AHB.

This is used to quickly write to the FIFO and free the AHB for transactions to peripherals other than the FMC, while the FMC is draining its FIFO into the memory. One of these register bits indicates the status of the FIFO, for ECC purposes.

The ECC is calculated while the data are written to the memory. To read the correct ECC, the software must consequently wait until the FIFO is empty.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																										FE	IF	IL	IR	IFS	ILS	IRS
																										r	rw	rw	rw	rw	rw	rw

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

- Bit 6 **FEMPT**: FIFO empty.
Read-only bit that provides the status of the FIFO
0: FIFO not empty
1: FIFO empty
- Bit 5 **IFEN**: Interrupt falling edge detection enable bit
0: Interrupt falling edge detection request disabled
1: Interrupt falling edge detection request enabled
- Bit 4 **ILEN**: Interrupt high-level detection enable bit
0: Interrupt high-level detection request disabled
1: Interrupt high-level detection request enabled
- Bit 3 **IREN**: Interrupt rising edge detection enable bit
0: Interrupt rising edge detection request disabled
1: Interrupt rising edge detection request enabled
- Bit 2 **IFS**: Interrupt falling edge status
The flag is set by hardware and reset by software.
0: No interrupt falling edge occurred
1: Interrupt falling edge occurred
- Bit 1 **ILS**: Interrupt high-level status
The flag is set by hardware and reset by software.
0: No Interrupt high-level occurred
1: Interrupt high-level occurred
- Bit 0 **IRS**: Interrupt rising edge status
The flag is set by hardware and reset by software.
0: No interrupt rising edge occurred
1: Interrupt rising edge occurred

Common memory space timing register 2..4 (FMC_PMEM2..4)

Address offset: Address: $0x48 + 0x20 * (x - 1)$, $x = 2..4$

Reset value: 0xFCFC FCFC

Each FMC_PMEMx ($x = 2..4$) read/write register contains the timing information for PC Card or NAND Flash memory bank x. This information is used to access either the common memory space of the 16-bit PC Card/CompactFlash, or the NAND Flash for command, address write access and data read/write access.

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

Bits 31:24 **MEMHIZx**: Common memory x data bus Hi-Z time

Defines the number of HCLK clock cycles during which the data bus is kept Hi-Z after the start of a PC Card/NAND Flash write access to common memory space on socket x. This is only valid for write transactions:

0000 0000: (0x00) 0 HCLK cycle (for PC Card) / 1 HCLK cycle (for NAND Flash)

1111 1111: (0xFF) 255 HCLK cycles (for PC Card) / 256 HCLK cycles (for NAND Flash)

Bits 23:16 **MEMHOLDx**: Common memory x hold time

Defines the number of HCLK clock cycles during which the address is held (and data for write accesses) after the command is deasserted (NWE, NOE), for PC Card/NAND Flash read or write access to common memory space on socket x:

0000 0000: reserved
 0000 0001: 1 HCLK cycle
 1111 1111: 255 HCLK cycles

Bits 15:8 **MEMWAITx**: Common memory x wait time

Defines the minimum number of HCLK (+1) clock cycles to assert the command (NWE, NOE), for PC Card/NAND Flash read or write access to common memory space on socket x. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:

0000 0000: reserved
 0000 0001: 2HCLK cycles (+ wait cycle introduced by deasserting NWAIT)
 1111 1111: 256 HCLK cycles (+ wait cycle introduced by the Card deasserting NWAIT)

Bits 7:0 **MEMSETx**: Common memory x setup time

Defines the number of HCLK (+1) clock cycles to set up the address before the command assertion (NWE, NOE), for PC Card/NAND Flash read or write access to common memory space on socket x:

0000 0000: 1 HCLK cycle (for PC Card) / HCLK cycles (for NAND Flash)
 1111 1111: 256 HCLK cycles (for PC Card) / 257 HCLK cycles (for NAND Flash)

Attribute memory space timing registers 2..4 (FMC_PATT2..4)

Address offset: $0x4C + 0x20 * (x - 1)$, $x = 2..4$

Reset value: 0xFCFC FCFC

Each FMC_PATTx ($x = 2..4$) read/write register contains the timing information for PC Card/CompactFlash or NAND Flash memory bank x. It is used for 8-bit accesses to the attribute memory space of the PC Card/CompactFlash or to access the NAND Flash for the last address write access if the timing must differ from that of previous accesses (for Ready/Busy management, refer to [Section 7.6.5: NAND Flash prewait functionality](#)).

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
ATTTHIZx								ATTTHOLDx								ATTWAITx								ATTSETx							
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bits 31:24 **ATTTHIZx**: Attribute memory x data bus Hi-Z time

Defines the number of HCLK clock cycles during which the data bus is kept in Hi-Z after the start of a PC CARD/NAND Flash write access to attribute memory space on socket x. Only valid for write transaction:

0000 0000: 0 HCLK cycle
 1111 1111: 255 HCLK cycles

Bits 23:16 **ATTTHOLDx**: Attribute memory x hold time

Defines the number of HCLK clock cycles during which the address is held (and data for write access) after the command deassertion (NWE, NOE), for PC Card/NAND Flash read or write access to attribute memory space on socket x

0000 0000: reserved
 0000 0001: 1 HCLK cycle
 1111 1111: 255 HCLK cycles

Bits 15:8 **ATTWAITx**: Attribute memory x wait time

Defines the minimum number of HCLK (+1) clock cycles to assert the command (NWE, NOE), for PC Card/NAND Flash read or write access to attribute memory space on socket x. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:

0000 0000: reserved

0000 0001: 2 HCLK cycles (+ wait cycle introduced by deassertion of NWAIT)

1111 1111: 256 HCLK cycles (+ wait cycle introduced by the card deasserting NWAIT)

Bits 7:0 **ATTSETx**: Attribute memory x setup time

Defines the number of HCLK (+1) clock cycles to set up address before the command assertion (NWE, NOE), for PC CARD/NAND Flash read or write access to attribute memory space on socket x:

0000 0000: 1 HCLK cycle

1111 1111: 256 HCLK cycles

I/O space timing register 4 (FMC_PIO4)

Address offset: 0xB0

Reset value: 0xFCFCFCFC

The FMC_PIO4 read/write registers contain the timing information used to access the I/O space of the 16-bit PC Card/CompactFlash.

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
IOHIZx								IOHOLDx								IOWAITx								IOSETx							
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	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 **IOHIZx**: I/O x data bus Hi-Z time

Defines the number of HCLK clock cycles during which the data bus is kept in Hi-Z after the start of a PC Card write access to I/O space on socket x. Only valid for write transaction:

0000 0000: 0 HCLK cycle

1111 1111: 255 HCLK cycles

Bits 23:16 IOHOLDx: I/O x hold time

Defines the number of HCLK clock cycles during which the address is held (and data for write access) after the command deassertion (NWE, NOE), for PC Card read or write access to I/O space on socket x:

0000 0000: reserved

0000 0001: 1 HCLK cycle

1111 1111: 255 HCLK cycles

Bits 15:8 IOWAITx: I/O x wait time

Defines the minimum number of HCLK (+1) clock cycles to assert the command (SMNWE, SMNOE), for PC Card read or write access to I/O space on socket x. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:

0000 0000: reserved, do not use this value

0000 0001: 2 HCLK cycles (+ wait cycle introduced by deassertion of NWAIT)

1111 1111: 256 HCLK cycles (+ wait cycle introduced by the Card deasserting NWAIT)

Bits 7:0 IOSETx: I/O x setup time

Defines the number of HCLK (+1) clock cycles to set up the address before the command assertion (NWE, NOE), for PC Card read or write access to I/O space on socket x:

0000 0000: 1 HCLK cycle

1111 1111: 256 HCLK cycles

ECC result registers 2/3 (FMC_ECCR2/3)

Address offset: $0x54 + 0x20 * (x - 1)$, $x = 2$ or 3

Reset value: 0x0000 0000

These registers contain the current error correction code value computed by the ECC computation modules of the FMC controller (one module per NAND Flash memory bank). When the CPU reads the data from a NAND Flash memory page at the correct address (refer to [Section 7.6.6: Computation of the error correction code \(ECC\) in NAND Flash memory](#)), the data read/written from/to the NAND Flash memory are processed automatically by the ECC computation module. When X bytes have been read (according to the ECCPS field in the FMC_PCRx registers), the CPU must read the computed ECC value from the FMC_ECCx registers. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and, to correct it otherwise. The FMC_ECCRx registers should be cleared after being read by setting the ECCEN bit to '0'. To compute a new data block, the ECCEN bit must be set to '1'.

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

Bits 31:0 **ECCx**: ECC result

This field contains the value computed by the ECC computation logic. [Table 51](#) describes the contents of these bit fields.

Table 51. ECC result relevant bits

ECCPS[2:0]	Page size in bytes	ECC bits
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

7.7 FMC register map

The following table summarizes the FMC registers.

Table 52. FMC register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FMC_BCR1	Reserved												CCLKEN	CBURSTW	Reserved		ASYNCSWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0x08	FMC_BCR2	Reserved												CBURSTW	CBURSTW	Reserved		ASYNCSWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0x10	FMC_BCR3	Reserved												CBURSTW	CBURSTW	Reserved		ASYNCSWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0x18	FMC_BCR4	Reserved												CBURSTW	CBURSTW	Reserved		ASYNCSWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0x04	FMC_BTR1	Res.	ACCMOD	DATLAT		CLKDIV		BUSTURN		DATAST				ADDHLD				ADDSET															
0x0C	FMC_BTR2	Res.	ACCMOD	DATLAT		CLKDIV		BUSTURN		DATAST				ADDHLD				ADDSET															
0x14	FMC_BTR3	Res.	ACCMOD	DATLAT		CLKDIV		BUSTURN		DATAST				ADDHLD				ADDSET															
0x1C	FMC_BTR4	Res.	ACCMOD	DATLAT		CLKDIV		BUSTURN		DATAST				ADDHLD				ADDSET															
0x104	FMC_BWTR1	Res.	ACCMOD	Res.				Reserved				DATAST				ADDHLD				ADDSET													
0x10C	FMC_BWTR2	Res.	ACCMOD	Res.				Reserved				DATAST				ADDHLD				ADDSET													
0x114	FMC_BWTR3	Res.	ACCMOD	Res.				Reserved				DATAST				ADDHLD				ADDSET													
0x11C	FMC_BWTR4	Res.	ACCMOD	Res.				Reserved				DATAST				ADDHLD				ADDSET													
0x60	FMC_PCR2	Reserved												ECCPS		TAR		TCLR		Res.	ECCEN	PWID		PTYP	PBKEN	PWAITEN	Reserved						
0x80	FMC_PCR3	Reserved												ECCPS		TAR		TCLR		Res.	ECCEN	PWID		PTYP	PBKEN	PWAITEN	Reserved						
0xA0	FMC_PCR4	Reserved												ECCPS		TAR		TCLR		Res.	ECCEN	PWID		PTYP	PBKEN	PWAITEN	Reserved						
0x64	FMC_SR2	Reserved																								EMPT	IFEN	ILEN	IREN	IFS	ILS	IRS	
0x84	FMC_SR3	Reserved																								EMPT	IFEN	ILEN	IREN	IFS	ILS	IRS	
0xA4	FMC_SR4	Reserved																								EMPT	IFEN	ILEN	IREN	IFS	ILS	IRS	

Table 52. FMC register map (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
0x68	FMC_PMEM2	MEMHIZx								MEMHOLDx								MEMWAITx								MEMSETx							
0x88	FMC_PMEM3	MEMHIZx								MEMHOLDx								MEMWAITx								MEMSETx							
0xA8	FMC_PMEM4	MEMHIZx								MEMHOLDx								MEMWAITx								MEMSETx							
0x6C	FMC_PATT2	ATTHIZx								ATTHOLDx								ATTWAITx								ATTSETx							
0x8C	FMC_PATT3	ATTHIZx								ATTHOLDx								ATTWAITx								ATTSETx							
0xAC	FMC_PATT4	ATTHIZx								ATTHOLDx								ATTWAITx								ATTSETx							
0xB0	FMC_PIO4	IOHIZx								IOHOLDx								IOWAITx								IOSETx							
0x74	FMC_ECCR2	ECCx																															
0x94	FMC_ECCR3	ECCx																															

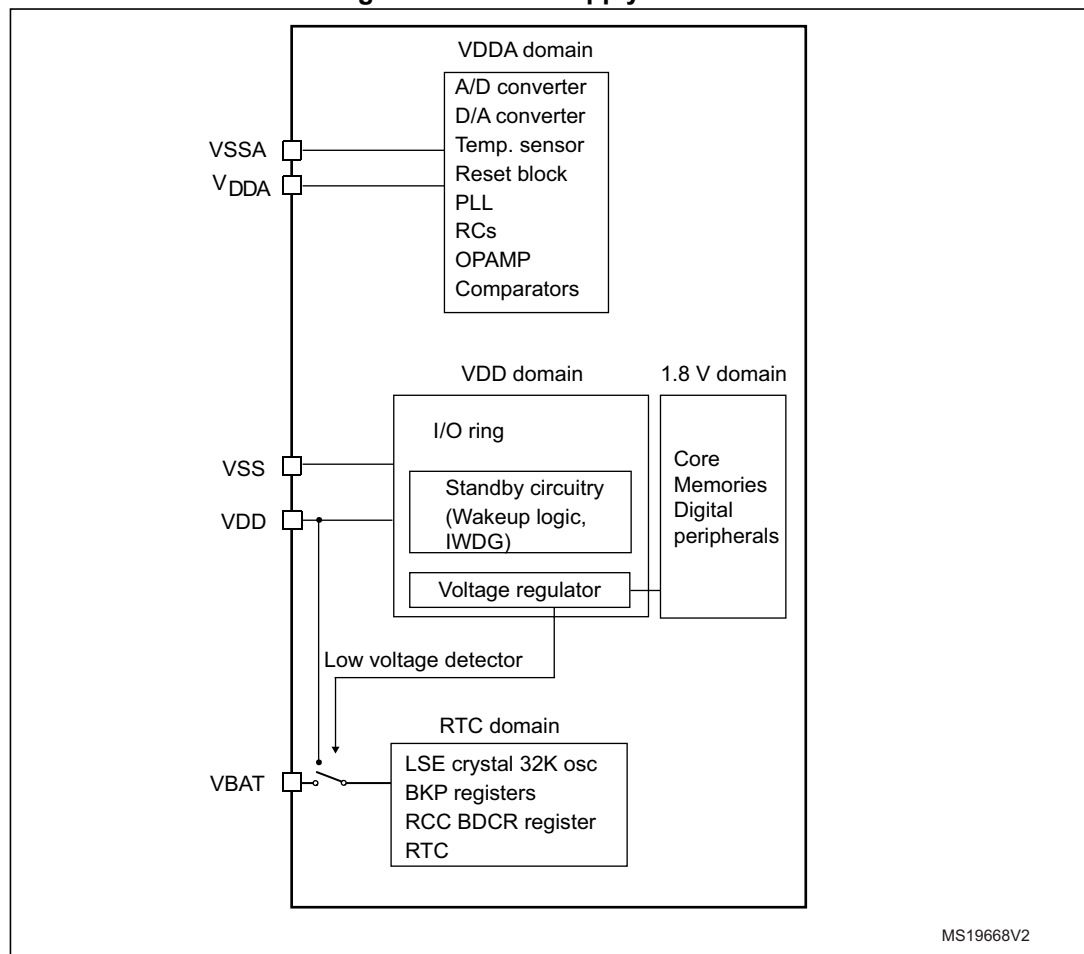
8 Power control (PWR)

8.1 Power supplies

STM32F302xx devices require a 2.0 V - 3.6 V operating supply voltage (V_{DD}) and a 2.0 V - 3.6 V analog supply voltage (V_{DDA}). The embedded regulator is used to supply the internal 1.8 V digital power.

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

Figure 30. Power supply overview



The following supply voltages are available:

- V_{DD} and V_{SS} : external power supply for I/Os and core.
These supply voltages are provided externally through V_{DD} and V_{SS} pins. V_{DD} = 2.0 to

- 3.6 V (devices).
 V_{DD} must always be kept lower than or equal to V_{DDA} .
- $V_{DD18} = 1.65$ to 1.95 V (V_{DD18} domain): power supply for digital core, SRAM and Flash memory.
 V_{DD18} is internally generated through an internal voltage regulator ().
 - V_{DDA} , $V_{SSA} = 2.0$ to 3.6 V (): external power supply for ADC, DAC, comparators, operational amplifiers, temperature sensor, PLL, HSI 8 MHz oscillator, LSI 40 kHz oscillator, and reset block.
 V_{DDA} must be in the 2.4 to 3.6 V range when the OPAMP and DAC are used.
It is forbidden to have $V_{DDA} < V_{DD} - 0.4$ V. An external Schottky diode must be placed between V_{DD} and V_{DDA} to guarantee that this condition is met.
 - $V_{BAT} = 1.65$ to 3.6 V: Backup power supply for RTC, LSE oscillator, PC13 to PC15 and backup registers when V_{DD} is not present. When V_{DD} supply is present, the internal power switch switches the backup power to V_{DD} . If V_{BAT} is not used, it must be connected to V_{DD} .

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

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

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

144- and 100-pin package connections (only on STM32F302xB/CD/E)

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

64-pin, 49-pin, 48-pin and 32-pin package connections

On these packages, the V_{REF+} and V_{REF-} pins are not available. They are internally connected to the ADC voltage supply (V_{DDA}) and ground (V_{SSA}) respectively.

The V_{DDA} supply/reference voltage can be equal to or higher than V_{DD} . When a single supply is used, V_{DDA} can be externally connected to V_{DD} , through the external filtering circuit in order to ensure a noise free V_{DDA} /reference voltage.

When V_{DDA} is different from V_{DD} , V_{DDA} must always be higher or equal to V_{DD} . To maintain a safe potential difference between V_{DDA} and V_{DD} during power-up/power-down, an external Schottky diode can be used between V_{DD} and V_{DDA} . Refer to the datasheet for the maximum allowed difference.

8.1.2 Battery backup domain

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

The V_{BAT} pin powers the RTC unit, the LSE oscillator and the PC13 to PC15 I/Os, allowing the RTC 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 is detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT} .
 During the startup phase, if V_{DD} is established in less than t_{RSTTEMPO} (Refer to the datasheet for the value of t_{RSTTEMPO}) and $V_{\text{DD}} > V_{\text{BAT}} + 0.6 \text{ V}$, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}).
 If the power supply/battery connected to the V_{BAT} pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the V_{BAT} pin.

If no external battery is used in the application, it is recommended to connect V_{BAT} externally to V_{DD} with a 100 nF external ceramic decoupling capacitor (for more details refer to AN4206).

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

- PC13, PC14 and PC15 can be used as GPIO pins
- PC13, PC14 and PC15 can be configured by RTC or LSE (refer to [Section 27.3: RTC functional description on page 724](#))

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

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

- PC13, PC14 and PC15 can be controlled only by RTC or LSE (refer to [Section 27.3: RTC functional description on page 724](#))

8.1.3 Voltage regulator

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

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

8.2 Power supply supervisor

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

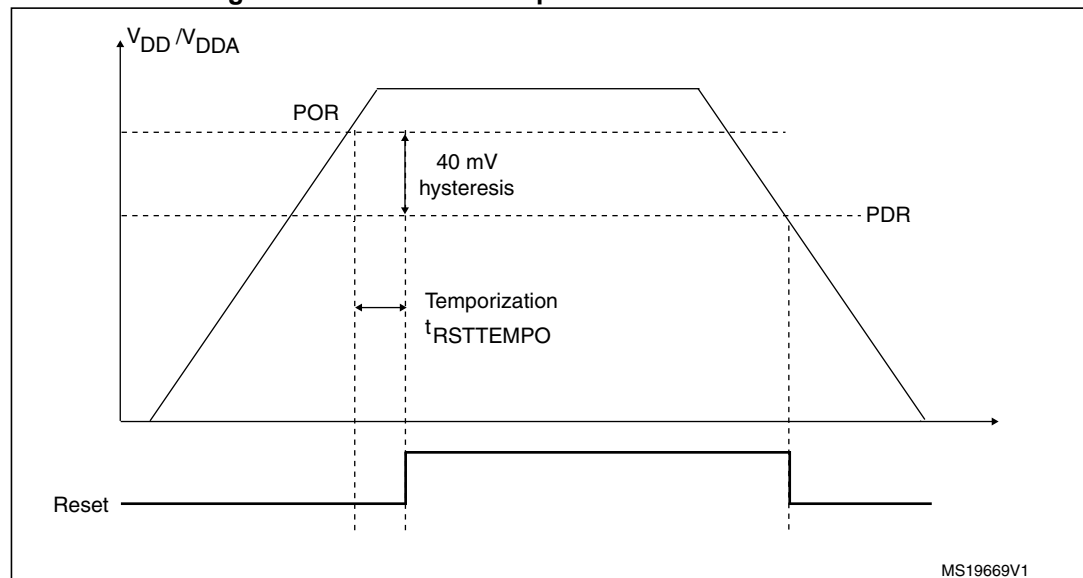
The device has an integrated power-on reset (POR) and power-down reset (PDR) circuits which are always active and ensure proper operation above a threshold of 2 V.

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

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

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

Figure 31. Power on reset/power down reset waveform



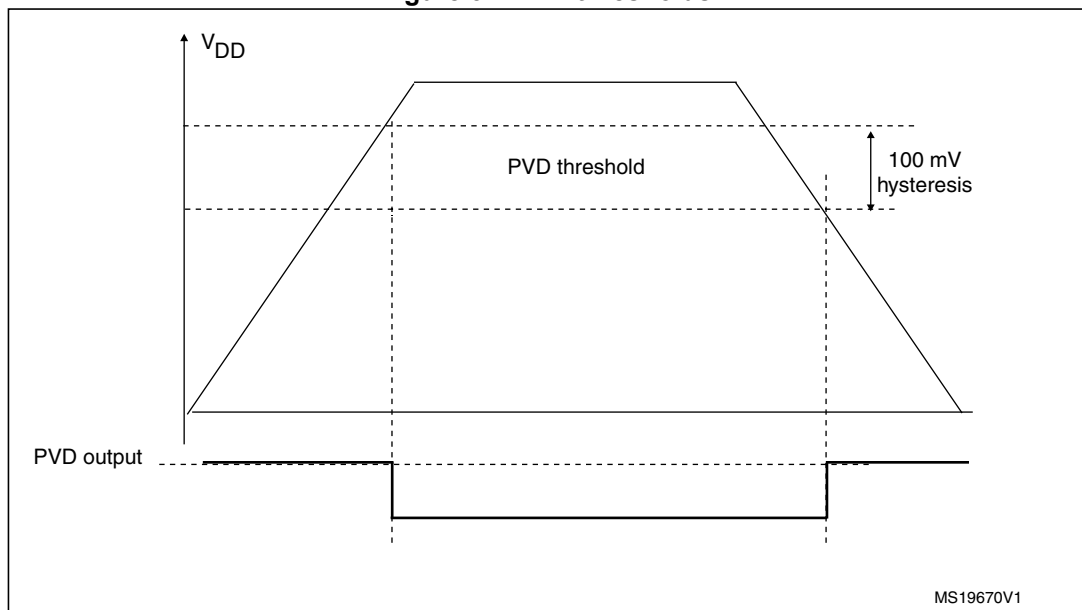
8.2.2 Programmable voltage detector (PVD)

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

The PVD is enabled by setting the PVDE bit.

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

Figure 32. PVD thresholds



8.3 Low-power modes

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

The device features three low-power modes:

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

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

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

Table 53. Low-power mode summary

Mode name	Entry	wakeup	Effect on 1.8V domain clocks	Effect on V _{DD} domain clocks	Voltage regulator
Sleep (Sleep now or Sleep-on - exit)	WFI	Any interrupt	CPU clock OFF no effect on other clocks or analog clock sources	None	ON
	WFE	Wakeup event			
Stop	PDDS and LPDS bits + SLEEPDEEP bit + WFI or WFE	Any EXTI line (configured in the EXTI registers) Specific communication peripherals on reception events (USART, I2C)	All 1.8V domain clocks OFF	HSI and HSE oscillators OFF	ON or in low-power mode (depends on Power control register (PWR_CR))
Standby	PDDS bit + SLEEPDEEP bit + WFI or WFE	WKUP pin rising edge, RTC alarm, external reset in NRST pin, IWDG reset			OFF

8.3.1 Slowing down system clocks

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

For more details refer to [Section 10.4.2: Clock configuration register \(RCC_CFGR\)](#).

8.3.2 Peripheral clock gating

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

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

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

8.3.3 Sleep mode

Entering Sleep mode

The Sleep mode is entered by executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the Cortex-M4[®]F System Control register:

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

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

Refer to [Table 54](#) and [Table 55](#) for details on how to enter Sleep mode.

Exiting Sleep mode

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

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

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex-M4[®]F System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

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

Refer to [Table 54](#) and [Table 55](#) for more details on how to exit Sleep mode.

Table 54. Sleep-now

Sleep-now mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 0 Refer to the Cortex-M4 [®] F System Control register.
Mode exit	If WFI was used for entry: Interrupt: Refer to Table 77: STM32F302xB/C/D/E vector table and Table 78: STM32F302x6/8 vector table . If WFE was used for entry Wakeup event: Refer to Section 14.2.3: Wakeup event management
Wakeup latency	None

Table 55. Sleep-on-exit

Sleep-on-exit	Description
Mode entry	WFI (wait for interrupt) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Refer to the Cortex-M4 [®] F System Control register.
Mode exit	Interrupt: refer to Table 77: STM32F302xB/C/D/E vector table and Table 78: STM32F302x6/8 vector table .
Wakeup latency	None

8.3.4 Stop mode

The Stop mode is based on the Cortex-M4[®]F deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode in the STM32F302xx devices. In the Stop mode, all I/O pins keep the same state as in the Run mode.

Entering Stop mode

Refer to [Table 56](#) for details on how to enter the Stop mode.

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

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

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

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a Reset. See

Section 26.3: IWDG functional description in *Section 26: Independent watchdog (IWDG)*.

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

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

Exiting Stop mode

Refer to *Table 56* for more details on how to exit Stop mode.

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

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

Table 56. Stop mode

Stop mode	Description
Mode entry	<p>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> – Set SLEEPDEEP bit in Cortex-M4[®]F System Control register – Clear PDDS bit in Power Control register (PWR_CR) – Select the voltage regulator mode by configuring LPDS bit in PWR_CR <p>Note: To enter Stop mode, all EXTI Line pending bits (in Pending register (EXTI_PR1)), all peripherals interrupt pending bits and RTC Alarm flag must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.</p> <p>If the application needs to disable the external oscillator (external clock) before entering Stop mode, the system clock source must be first switched to HSI and then clear the HSEON bit.</p> <p>Otherwise, if before entering Stop mode the HSEON bit is kept at 1, the security system (CSS) feature must be enabled to detect any external oscillator (external clock) failure and avoid a malfunction when entering Stop mode.</p>
Mode exit	<p>If WFI was used for entry:</p> <ul style="list-style-type: none"> – Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). – Some specific communication peripherals (USART, I2C) interrupts, when programmed in wakeup mode (the peripheral must be programmed in wakeup mode and the corresponding interrupt vector must be enabled in the NVIC). <p>Refer to Table 77: STM32F302xB/C/D/E vector table and Table 78: STM32F302x6/8 vector table.</p> <p>If WFE was used for entry:</p> <p>Any EXTI Line configured in event mode. Refer to Section 14.2.3: Wakeup event management on page 274</p>
Wakeup latency	HSI RC wakeup time + regulator wakeup time from Low-power mode

8.3.5 Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the Cortex-M4[®]F deepsleep mode, with the voltage regulator disabled. The 1.8 V domain is consequently powered off. The PLL, the HSI oscillator and the HSE oscillator are also switched off. SRAM and register contents are lost except for registers in the RTC domain and Standby circuitry (see [Figure 30](#)).

Entering Standby mode

Refer to [Table 57](#) for more details on how to enter Standby mode.

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a reset. See

[Section 26.3: IWDG functional description](#) in [Section 26: Independent watchdog \(IWDG\)](#).

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

Exiting Standby mode

The microcontroller exits the Standby mode when an external reset (NRST pin), an IWDG reset, a rising edge on the WKUP pin or the rising edge of an RTC alarm occurs (see [Figure 282: RTC block diagram](#)). All registers are reset after wakeup from Standby except for [Power control/status register \(PWR_CSR\)](#).

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

Refer to [Table 57](#) for more details on how to exit Standby mode.

Table 57. Standby mode

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> – Set SLEEPDEEP in Cortex-M4[®]F System Control register – Set PDDS bit in Power Control register (PWR_CR) – Clear WUF bit in Power Control/Status register (PWR_CSR)
Mode exit	WKUP pin rising edge, RTC alarm event's rising edge, external Reset in NRST pin, IWDG Reset.
Wakeup latency	Reset phase

I/O states in Standby mode

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

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

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the Cortex-M4[®]F core is no longer clocked.

However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged even when using the low-power modes extensively.

8.3.6 Auto-wakeup from low-power mode

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

- Low-power 32.768 kHz external crystal oscillator (LSE OSC).
This clock source provides a precise time base with very low-power consumption (less than 1µA added consumption in typical conditions)
- Low-power internal RC Oscillator (LSI RC)
This clock source has the advantage of saving the cost of the 32.768 kHz crystal. This internal RC Oscillator is designed to add minimum power consumption.

To wakeup from Stop mode with an RTC alarm event, it is necessary to:

- Configure the EXTI Line 17 to be sensitive to rising edge
- Configure the RTC to generate the RTC alarm

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

8.4 Power control registers

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

8.4.1 Power control register (PWR_CR)

Address offset: 0x00

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

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

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

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

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

0: Access to RTC and Backup registers disabled

1: Access to RTC and Backup registers enabled

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

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

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

000: 2.2V

001: 2.3V

010: 2.4V

011: 2.5V

100: 2.6V

101: 2.7V

110: 2.8V

111: 2.9V

Notes:

1. Refer to the electrical characteristics of the datasheet for more details.
2. Once the PVD_LOCK is enabled (for CLASS B protection) the PLS[2:0] bits cannot be programmed anymore.

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

This bit is set and cleared by software.

0: PVD disabled

1: PVD enabled

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

This bit is always read as 0.

0: No effect

1: Clear the SBF Standby Flag (write).

- Bit 2 **CWUF**: Clear wakeup flag.
This bit is always read as 0.
0: No effect
1: Clear the WUF Wakeup Flag **after 2 System clock cycles**. (write)
- Bit 1 **PDDS**: Power down deepsleep.
This bit is set and cleared by software. It works together with the LPDS bit.
0: Enter Stop mode when the CPU enters Deepsleep. The regulator status depends on the LPDS bit.
1: Enter Standby mode when the CPU enters Deepsleep.
- Bit 0 **LPDS**: Low-power deepsleep.
This bit is set and cleared by software. It works together with the PDDS bit.
0: Voltage regulator on during Stop mode
1: Voltage regulator in low-power mode during Stop mode

8.4.2 Power control/status register (PWR_CSR)

Address offset: 0x04

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	EWUP3	EWUP2	EWUP1	Res	Res	Res	Res	Res	PVDO	SBF	WUF
					rw	rw	rw						r	r	r

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

- Bit 10 **EWUP3**: Enable WKUP3 pin
This bit is set and cleared by software.
0: WKUP3 pin is used for general purpose I/O. An event on the WKUP3 pin does not wakeup the device from Standby mode.
1: WKUP3 pin is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP3 pin wakes-up the system from Standby mode).

Note: This bit is reset by a system Reset.

- Bit 9 **EWUP2**: Enable WKUP2 pin
This bit is set and cleared by software.
0: WKUP2 pin is used for general purpose I/O. An event on the WKUP2 pin does not wakeup the device from Standby mode.
1: WKUP2 pin is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP2 pin wakes-up the system from Standby mode).

Note: This bit is reset by a system Reset.

Bit 8 **EWUP1**: Enable WKUP1 pin

This bit is set and cleared by software.

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

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

Note: This bit is reset by a system Reset.

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

Bit 2 **PVDO**: PVD output

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

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

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

Notes:

1. The PVD is stopped by Standby mode. For this reason, this bit is equal to 0 after Standby or reset until the PVDE bit is set.
2. Once the PVD is enabled and configured in the PWR_CR register, PVDO can be used to generate an interrupt through the External Interrupt controller.
3. Once the PVD_LOCK is enabled (for CLASS B protection) PVDO cannot be disabled anymore.

Bit 1 **SBF**: Standby flag

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

0: Device has not been in Standby mode

1: Device has been in Standby mode

Bit 0 **WUF**: Wakeup flag

This bit is set by hardware and cleared by a system reset or by setting the CWUF bit in the [Power control register \(PWR_CR\)](#)

0: No wakeup event occurred

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

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

8.4.3 PWR register map

The following table summarizes the PWR registers.

Table 58. PWR register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x000	PWR_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Reset value																								0	0	0	0	0	0	0	0	0	
0x004	PWR_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWUP3	EWUP2	EWUP1	Res.	Res.	Res.	Res.	Res.	PVDO	SBF	WUF	
	Reset value																					0	0	0						0	0	0		

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

9 Peripheral interconnect matrix

9.1 Introduction

Several STM32F3 peripherals have internal interconnections. Knowing these interconnections allows the following benefits:

- Autonomous communication between peripherals,
- Efficient synchronization between peripherals,
- Discard the software latency and minimize GPIOs configuration,
- Optimum number of available pins even with small packages,
- Avoid the use of connectors and design an optimized PCB with less dissipated energy.

9.2 Connection summary

The following table presents the matrix for the peripheral interconnect.

Table 59. STM32F302xx peripherals interconnect matrix⁽¹⁾

Source	Destination																		
	DMA1	DMA2 (2)	ADC1	ADC2 (2)	COMP1(2)	COMP2	COMP4	COMP6	OPAMP1(2)	OPAMP2	TIM1	TIM15	TIM16	TIM17	TIM2	TIM3(2)	TIM4(2)	DAC1	IRTIM
ADC1	x	-	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ADC2(2)	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP1(2)	-	-	-	-	-	-	-	-	-	-	x	-	-	-	x	x	-	-	-
COMP2	-	-	-	-	-	-	-	-	-	-	x	-	-	-	x	x	-	-	-
COMP4	-	-	-	-	-	-	-	-	-	-	-	x	-	-	-	x	x	-	-
COMP6	-	-	-	-	-	-	-	-	-	-	-	-	x	-	x	-	x	-	-
OPAMP1(2)	-	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OPAMP2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM3(2)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM4(2)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAC1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IRTIM	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 59. STM32F302xx peripherals interconnect matrix⁽¹⁾ (continued)

Source	Destination																		
	DMA1	DMA2 (2)	ADC1	ADC2 (2)	COMP1(2)	COMP2	COMP4	COMP6	OPAMP1(2)	OPAMP2	TIM1	TIM15	TIM16	TIM17	TIM2	TIM3(2)	TIM4(2)	DAC1	IRTIM
OPAMP2	-	-	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM1	x	-	x	x	x	x	-	-	x	x	-	-	-	-	x	x	x	-	-
SPI1(2)	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
USART1	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SPI4(3)	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM15	x	-	x	x	-	-	x	x	-	-	x	-	-	-	-	x	-	x	-
TIM16	x	-	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	x
TIM17	x	-	-	-	-	-	-	-	-	-	x	x	-	-	-	-	-	-	x
TIM2	x	-	x	x	x	x	-	x	-	-	x	x	-	-	-	x	x	x	-
TIM3(2)	x	-	x	x	x	x	x	-	-	-	x	x	-	-	x	-	x	x	-
TIM4(2)	x	-	x	x	-	-	-	-	-	-	x	-	-	-	x	x	-	x	-
TIM6	x	x	x	x	-	-	-	-	-	-	-	-	-	-	-	-	-	x	
SPI2/I2S	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SPI3/I2S	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
USART2	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 59. STM32F302xx peripherals interconnect matrix⁽¹⁾ (continued)

Source	Destination																		
	DMA1	DMA2 (2)	ADC1	ADC2 (2)	COMP1(2)	COMP2	COMP4	COMP6	OPAMP1(2)	OPAMP2	TIM1	TIM15	TIM16	TIM17	TIM2	TIM3(2)	TIM4(2)	DAC1	IRTIM
USART3	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UART4(2)	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UART5(2)	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
I2C1	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
I2C2	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAC1	x	x	-	-	x	x	x	x	x	-	-	-	-	-	-	-	-	-	-
I2C3(4)	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TS	-	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VBAT	-	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Vrefint	-	-	x	x	x	x	x	x	-	-	-	-	-	-	-	-	-	-	-
CSS	-	-	-	-	-	-	-	-	-	-	x	x	-	-	-	-	-	-	-
PVD	-	-	-	-	-	-	-	-	-	-	x	x	-	-	-	-	-	-	-
SRAM Parity error(2)	-	-	-	-	-	-	-	-	-	-	x	x	-	-	-	-	-	-	-
CPU Hardfault	-	-	-	-	-	-	-	-	-	-	x	x	-	-	-	-	-	-	-
HSE	-	-	-	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-
HSI	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 59. STM32F302xx peripherals interconnect matrix⁽¹⁾ (continued)

Source	Destination																		
	DMA1	DMA2 (2)	ADC1	ADC2 (2)	COMP1(2)	COMP2	COMP4	COMP6	OPAMP1(2)	OPAMP2	TIM1	TIM15	TIM16	TIM17	TIM2	TIM3(2)	TIM4(2)	DAC1	IRTIM
RTC	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	X	•	•	•	•	•	•
	•		•	•	•	•	•	•	•	•	•	•	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. X means interconnect, and “-” means no interconnect.

2. Only in STM32F302xB/C/D/E.

3. Only in STM32F302xD/E.

4. Not in STM32F302xB/C.

9.3 Interconnection details

9.3.1 DMA interconnections

Hardware DMA requests are managed by peripherals. The DMA channels dedicated to each peripheral are summarized in [Section 13.4.7: DMA request mapping](#).

9.3.2 From ADC to ADC

ADC1 can be used as a "master" to trigger ADC2 "slave" start of conversion.

In dual ADC mode, the converted data of the master and slave ADCs can be read in parallel.

A description of dual ADC mode is provided in [Section 15.3.29: Dual ADC modes \(STM32F302xB/C only\)](#).

9.3.3 From ADC to TIM

ADC1 can provide trigger event through watchdog signals to advanced-control timers (TIM1).

A description of the ADC analog watchdog settings is provided in [Section 15.3.28: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx\)](#).

The output (from ADC) is on signals ADC1_AWDx_OUT (x = 1..3 as there are 3 analog watchdogs per ADC) and the input (to timer) on signal TIMx_ETR (external trigger).

TIMx_ETR is connected to ADC1_AWDx_OUT through bits in TIM1_OR registers; refer to [Section 20.4.24: TIM1 option registers \(TIMx_OR\)](#).

9.3.4 From TIM and EXTI to ADC

General-purpose timers (TIM2/TIM3/TIM4), basic timers (TIM6), advanced-control timer (TIM1), general-purpose timer (TIM15/TIM16/TIM17) and EXTI can be used to generate an ADC triggering event.

The output (from timer) is on signal TIMx_TRGO, TIMx_TRGO2 or TIMx_CCx event.

The input (to ADC) is on signal EXT[15:0], JEXT[15:0].

The connection between timers and ADCs or also EXTI & ADCs is provided in:

- [Table 86: ADC1 \(master\) & 2 \(slave\) - External triggers for regular channels](#)
- [Table 87: ADC1 & ADC2 - External trigger for injected channels](#)

9.3.5 From OPAMP to ADC

There are two interconnection types:

1. Connect OPAMP output reference voltage to an internal ADC channel. This connection can be used for OPAMP calibration. For more details, please refer to the Section *Calibration* in the OPAMP chapter.

[Section 15.3.11: Channel selection \(SQRx, JSQRx\)](#) provides the exact ADC channels to be used.

Table 60. VREFOPAMPx to ADC channel

VREFOPAMPx	ADC channel
VREFOPAMP1 ⁽¹⁾	ADC1_IN15
VREFOPAMP2	ADC2_IN17 ⁽¹⁾

1. Only in STM32F302xB/C/D/E.

2. OPAMPx output, x = 1, 2 can be connected to ADCy channels, y = 1, 2 through the GPIOs. See summary in the table below. Refer to [Section 18.3.4: Using the OPAMP outputs as ADC inputs](#).

Table 61. OPAMP output to ADC input

OPAMPx output	ADC channel	Used pins
OPAMP1_VOUT ⁽¹⁾	ADC1_IN3	PA2
OPAMP2_VOUT	ADC2_IN3 ⁽¹⁾	PA6

1. Only in STM32F302xB/C/D/E.

9.3.6 From TS to ADC

Internal temperature sensor (VTS) is connected internally to ADC1_IN16. Refer to [Section 15.3.30: Temperature sensor](#).

9.3.7 From VBAT to ADC

VBAT/2 output voltage can be converted using ADC1_IN17. This interconnection is explained in [Section 15.3.31: VBAT supply monitoring](#).

9.3.8 From VREFINT to ADC

VREFINT is internally connected to channel 18 of the ADCs. This allows the monitoring of its value as described in [Section 15.3.32: Monitoring the internal voltage reference](#).

9.3.9 From COMP to TIM

The comparators outputs can be redirected internally to different timer inputs:

- break input 1/2 for fast PWM shutdowns,
- OCREF_CLR input,
- Input capture.

To select which timer input must be connected to the comparator output, the bits field COMPxOUTSEL in the COMPx_CSR register are used.

The following table gives an overview of all possible comparator outputs redirection to the timer inputs.

Table 62. Comparator outputs to timer inputs

COMP output selection						
	TIM1	TIM2	TIM3 ⁽¹⁾	TIM4	TIM15	TIM16
COMP ⁽¹⁾	TIM1_BRK_ACTH TIM1_BRK2 TIM1_OCrefClear TIM1_IC1	TIM2_IC4 TIM2_OCrefClear	TIM3_IC1 TIM3_OCrefClear	N.A	N.A	N.A
COMP2	TIM1_BRK_ACTH TIM1_BRK2 TIM1_OCrefClear TIM1_IC1	TIM2_IC4 TIM2_OCrefClear	TIM3_IC1 TIM3_OCrefClear	N.A	N.A	N.A

Table 62. Comparator outputs to timer inputs (continued)

COMP output selection						
	TIM1	TIM2	TIM3 ⁽¹⁾	TIM4	TIM15	TIM16
COMP4	TIM1_BRK TIM1_BRK2	N.A	TIM3_IC3 TIM3_OCrefClear	TIM4_IC2	TIM15_OCrefClear TIM15_IC2	N.A
COMP6	TIM1_BRK_ACTH TIM1_BRK2	TIM2_IC2 TIM2_OCrefClear	N.A	TIM4_IC4	N.A	TIM16_OCrefClear TIM16_IC1

1. Only in STM32F302xB/C/D/E.

Note: When the comparator output is configured to be connected internally to timers break input, the following must be considered:

1/ COMP1/2/6 can be used to control TIM1_BRK_ACTH (this break is always active high with no digital filter) and to control also TIM1_BRK2 input.

2/ COMP4 can be used to control TIM1_BRK and the TIM1_BRK2 input (same as the other comparators).

9.3.10 From TIM to COMP

The timers output can be selected as comparators outputs blanking signals using the “COMPx_BLANKING” bits in “COMPx_CSR” register. More details on the blanking function can be found in [Section 17.3.6: Comparator output blanking function](#).

Table 63. Timer output selection as comparator blanking source

COMP blanking source				
	COMP1	COMP2	COMP4	COMP6
TIM1	TIM1 OC5	TIM1 OC5	-	-
TIM15	-	-	TIM15 OC1	TIM15 OC2
TIM2	TIM2 OC3	TIM2 OC3	-	TIM2 OC4
TIM3	TIM3 OC3	TIM3 OC3	TIM3 OC4	-

9.3.11 From DAC to COMP

The comparators inverting input may be a DAC channel output (DAC1_CH1).

The selection is made based on “COMPxINMSEL” bits value in “COMPx_CSR” register.

9.3.12 From VREFINT to COMP

Besides to the DAC channel output, Vrefint (x1, x3/4, x1/2, x1/4) can be selected as comparator inverting input using “COMPxINMSEL” bits in “COMPx_CSR” register.

9.3.13 From DAC to OPAMP

In STM32F302xB/C/D/E, the DAC1 output is connected internally to OPAMP1 non-inverting input.

9.3.14 From TIM to OPAMP

The switch between OPAMP inverting and non-inverting inputs can be done automatically. This automatic switch is triggered by the TIM1 CC6 output arriving on the OPAMP input multiplexers. More details on this feature are available in [Section 18.3.6: Timer controlled Multiplexer mode](#).

9.3.15 From TIM to TIM

Some STM32F3 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 with the various synchronization modes is available in:

- [Section 20.3.24: Timer synchronization for the advanced-control timers \(TIM1\)](#)
- [Section 21.3.19: Timer synchronization for the general-purpose timers \(TIM2/TIM3/TIM4\)](#)

The slave mode selection is made using “SMS” bits, as described in:

- [Section 20.4.3: TIM1 slave mode control register \(TIMx_SMCR\)](#),
- [Section 21.4.3: TIMx slave mode control register \(TIMx_SMCR\) for the general-purpose timers \(TIM2/TIM3/TIM4\)](#),
- [Section 22.5.3: TIM15 slave mode control register \(TIM15_SMCR\)](#).

The possible master/slave connections are summarized in the following table providing the internal trigger connection:

Table 64. Timer synchronization

		SLAVE				
		TIM1	TIM2	TIM3 ⁽¹⁾	TIM4	TIM15
MASTER	TIM1	-	TIM2_ITR0	TIM3_ITR0	TIM4_ITR0	-
	TIM2	TIM1_ITR1	-	TIM3_ITR1	TIM4_ITR1	TIM15_ITR0
	TIM3	TIM1_ITR2	TIM2_ITR2	-	TIM4_ITR2	TIM15_ITR1
	TIM4	TIM1_ITR3	TIM2_ITR3	TIM3_ITR3	-	-
	TIM15	TIM1_ITR0	-	TIM3_ITR2	-	-
	TIM16	-	-	-	-	TIM15_ITR2
	TIM17	TIM1_ITR3	-	-	-	TIM15_ITR3

1. Only in STM32F302xB/C/D/E

9.3.16 From break input sources to TIM

In addition to comparators outputs, other sources can be used as trigger for the internal break events of some timers (TIM1/TIM15/TIM16/TIM17). For example:

- the clock failure event generated by CSS, refer to [Section 10.2.6: System clock \(SYSCLK\) selection](#) for more details,
- the PVD output, refer to [Section 8.2.2: Programmable voltage detector \(PVD\)](#) for more details,
- the SRAM parity error signal, refer to [Section 3.3.1: Parity check](#) for more details,
- the Cortex-M4 LOCKUP (Hardfault) output.

The sources mentioned above can be connected internally to TIMx_BRK_ACTH input, x = 1,15,16,17.

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

More details on the break feature are provided in:

- [Section 20.3.15: Using the break function](#) for the advanced-control timers (TIM1)
- [Section 22.4.13: Using the break function](#) for the general-purpose timers (TIM15/TIM16/TIM17)

9.3.17 From HSE, HSI, LSE, LSI, MCO, RTC to TIM

TIM16 can be used for the measurement of internal/external clock sources. TIM16 channel1 input capture is connected to HSE/32, GPIO, RTC clock and MCO to output clocks among (HSE, HSI, LSE, LSI, SYSCLK, PLLCLK, PLLCLK/2).

The selection is performed through the TI1_RMP [1:0] bits in the TIM16_OR register.

This allows calibrating the HSI/LSI clocks.

More details are provided in [Section 10.2.14: Internal/external clock measurement with TIM16](#).

9.3.18 From TIM and EXTI to DAC

A timer counter may be used as a trigger for DAC conversions.

The TRGO event is the internal signal that will trigger conversion.

The following table provides a summary of DACs interconnections with timers:

This is described in [Section 16.3.6: DAC trigger selection](#).

Table 65. Timer and EXTI signals triggering DAC1 conversions

Timer	DAC1 ⁽¹⁾
TIM2	X
TIM3⁽¹⁾	X
TIM4⁽¹⁾	X
TIM6	X
TIM15	X
EXTI line9	X

1. Only in STM32F302xB/C/D/E devices.

9.3.19 From TIM to IRTIM

General-purpose timer (TIM16/TIM17) output channels TIMx_OC1 are used to generate the waveform of infrared signal output. The functionality is described in [Section 24: Infrared interface \(IRTIM\)](#).

10 Reset and clock control (RCC)

10.1 Reset

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

10.1.1 Power reset

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

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

A power reset sets all registers to their reset values except the RTC domain (see [Figure 30 on page 147](#)).

10.1.2 System reset

A system reset sets all registers to their reset values except the reset flags in the clock controller CSR register and the registers in the RTC domain (see [Figure 30 on page 147](#)).

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

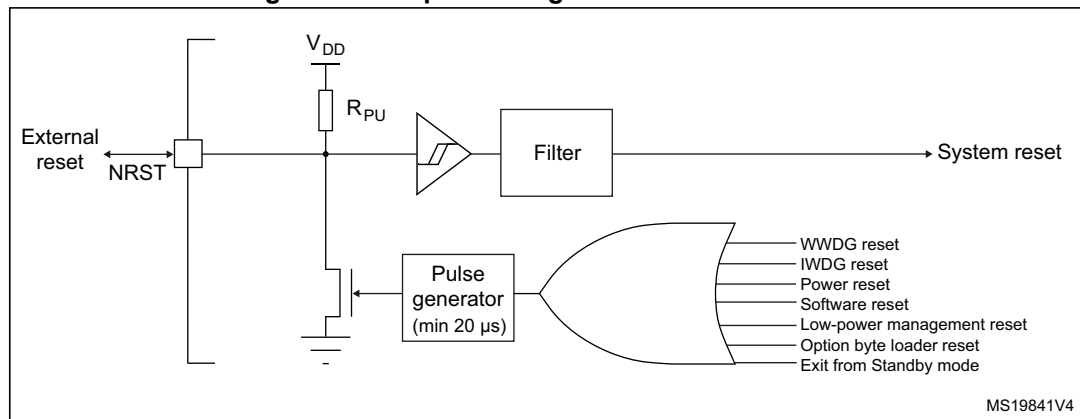
1. A low level on the NRST pin (external reset)
2. Window watchdog event (WWDG reset)
3. Independent watchdog event (IWDG reset)
4. A software reset (SW reset) (see [Software reset](#))
5. Low-power management reset (see [Low-power management reset](#))
6. Option byte loader reset (see [Option byte loader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

The reset source can be identified by checking the reset flags in the Control/Status register, RCC_CSR (see [Section 10.4.10: Control/status register \(RCC_CSR\)](#)).

These sources act on the NRST pin and it is always kept low during the delay phase. The RESET service routine vector is fixed at address 0x0000_0004 in the memory map.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 μ s for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

Figure 33. Simplified diagram of the reset circuit



Software reset

The SYSRESETREQ bit in Cortex-M4®F Application Interrupt and Reset Control Register must be set to force a software reset on the device. Refer to the STM32F3xx/F4xx Cortex®-M4 programming manual (PM0214) for more details.

Low-power management reset

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

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

For further information on the User Option Bytes, refer to [Section 4: Option bytes](#).

Option byte loader reset

The option byte loader reset is generated when the OBL_LAUNCH bit (bit 13) is set in the FLASH_CR register. This bit is used to launch the option byte loading by software.

10.1.3 RTC domain reset

The RTC domain has two specific resets that affect only the RTC domain ([Figure 30 on page 147](#)).

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

1. Software reset, triggered by setting the BDRST bit in the [RTC domain control register \(RCC_BDCR\)](#).
2. V_{DD} power-up if V_{BAT} has been disconnected when it was low.

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

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

10.2 Clocks

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

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

The devices have the following additional clock sources:

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

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

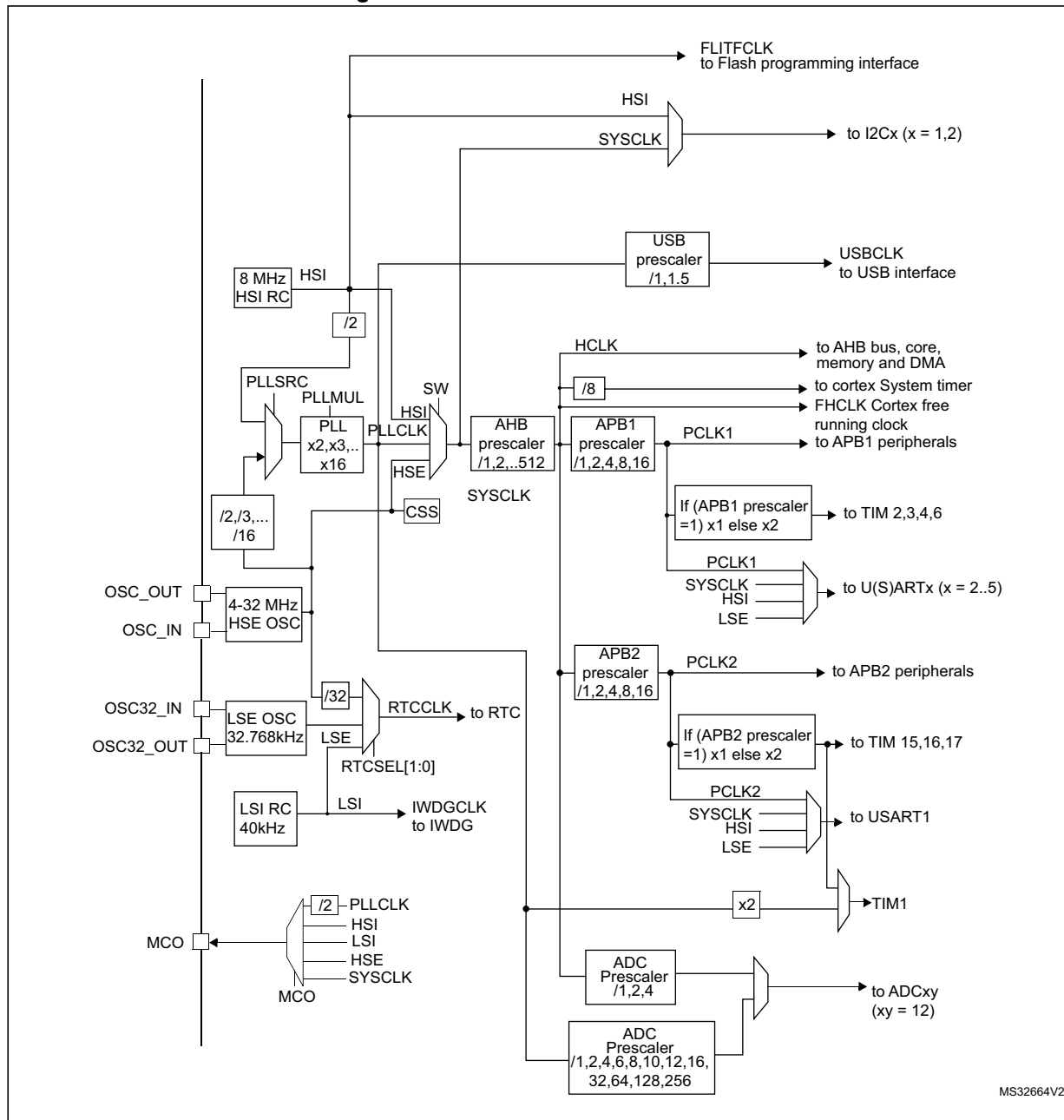
Several prescalers can be used to configure the AHB frequency, the high speed APB (APB2) and the low speed APB (APB1) domains. The maximum frequency of the AHB and APB2 domains is 72 MHz. The maximum allowed frequency of the APB1 domain is 36 MHz.

All the peripheral clocks are derived from their bus clock (HCLK, PCLK1 or PCLK2) except:

- The Flash memory programming interface clock (FLITFCLK) which is always the HSI clock.
- The 48-MHz USB clock which is derived from the PLL VCO
- The option byte loader clock which is always the HSI clock
- The ADCs clock which is derived from the PLL output. It can reach 72 MHz and can then be divided by 1,2,4,6,8,10,12,16,32,64,128 or 256.
- The U(S)ARTs clock which is derived (selected by software) from one of the four following sources:
 - system clock
 - HSI clock
 - LSE clock
 - APB1 or APB2 clock (PCLK1 or PCLK2 depending on which APB is mapped the)
- The I2C1/2 (I2C1/2/3 on STM32F302xD/E) clock which is derived (selected by software) from one of the two following sources:
 - system clock
 - HSI clock
- The I2S2 and I2S3 clocks which can be derived from an external dedicated clock source.
- The RTC clock which is derived from the LSE, LSI or from the HSE clock divided by 32.
- The IWDG clock which is always the LSI clock.

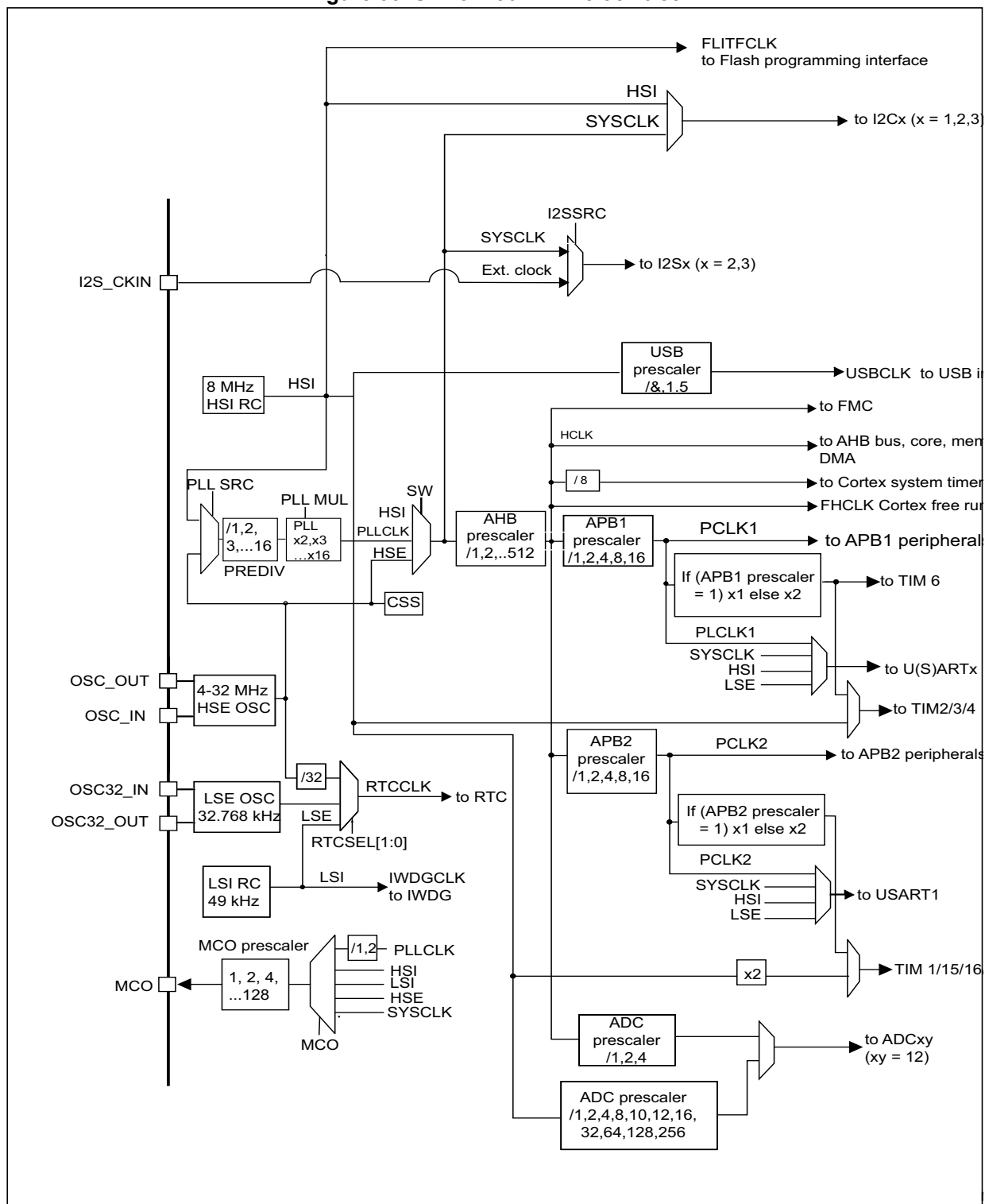
The RCC feeds the Cortex® System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or directly with the Cortex® clock (HCLK), configurable in the SysTick Control and Status Register.

Figure 34. STM32F302xB/C clock tree



1. For full details about the internal and external clock source characteristics, please refer to the "Electrical characteristics" section in your device datasheet.
2. TIM1 can be clocked from the PLLCLKx2 running up to 144 MHz when the system clock source is the PLL. Refer to [Section 10.2.10: Timers \(TIMx\) clock](#).
3. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). When the programmable factor is '1', the AHB prescaler must be equal to '1'.

Figure 35. STM32F302xD/E clock tree

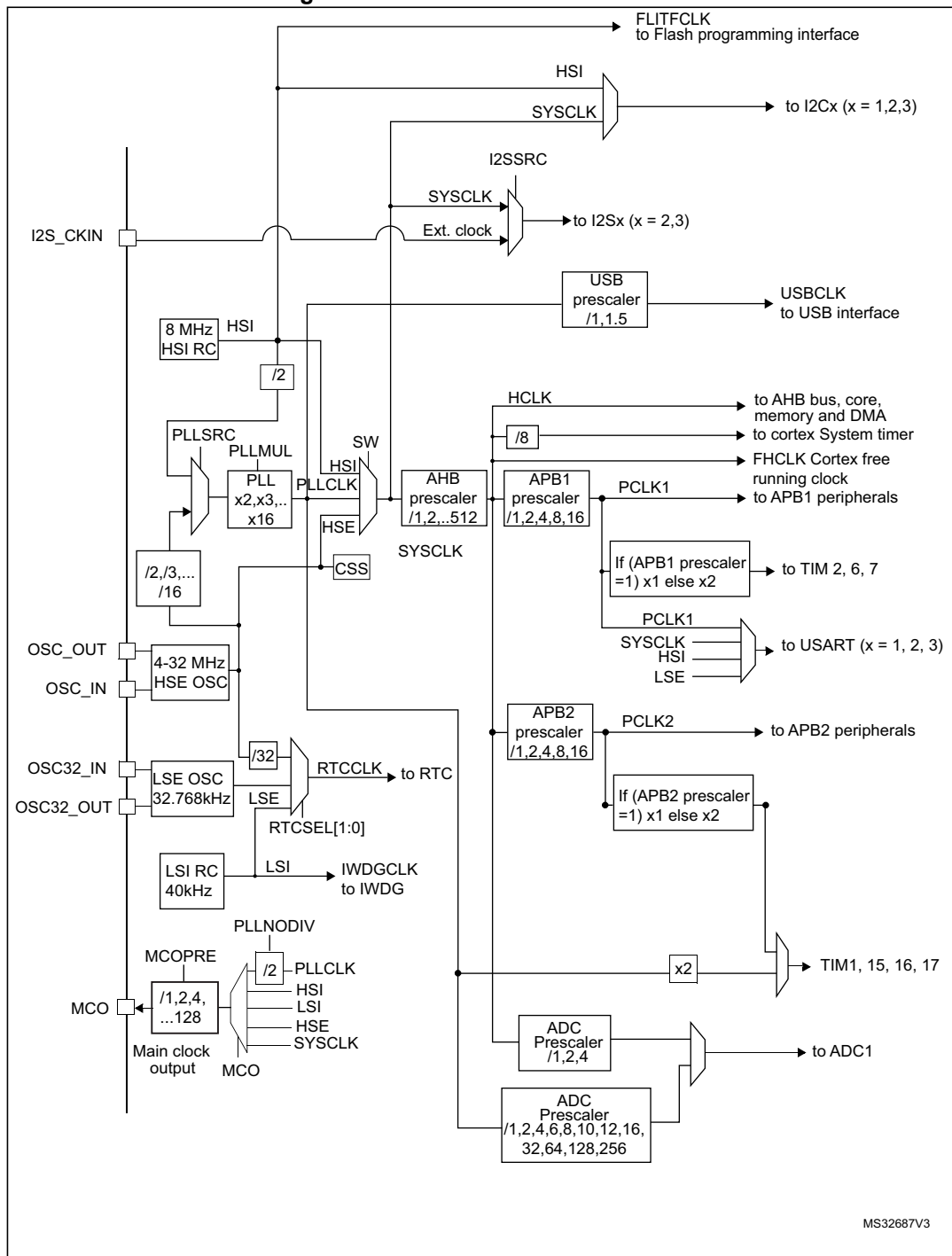


1. For full details about the internal and external clock source characteristics, please refer to the Electrical characteristics section in your device datasheet.
2. TIM1 can be clocked from the PLL running at 144 MHz when the system clock source is the PLL and AHB

or APB2 subsystem clocks are not divided by more than 2 cumulatively.

3. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). When the programmable factor is '1', the AHB prescaler must be equal to '1'.

Figure 36. STM32F302x6/8 clock tree



MS32687V3

1. For full details about the internal and external clock source characteristics, please refer to the “Electrical characteristics” section in your device datasheet.
2. TIM1 can be clocked from the PLL running at 144 MHz when the system clock source is the PLL and AHB or APB2

- subsystem clocks are not divided by more than 2 cumulatively.
3. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). When the programmable factor is '1', the AHB prescaler must be equal to '1'.

FCLK acts as Cortex-M4[®]F free-running clock. For more details refer to the STM32F3xx/F4xx Cortex[®]-M4 programming manual (PM0214).

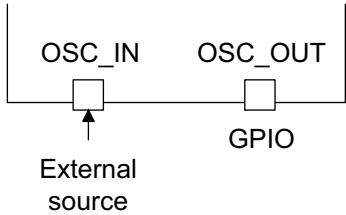
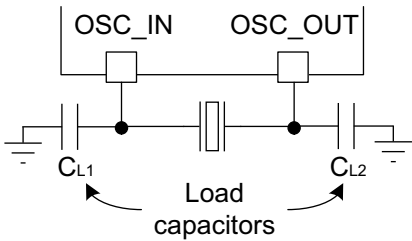
10.2.1 HSE clock

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 37. HSE/ LSE clock sources

Clock source	Hardware configuration
External clock	<div></div> <div>MSv31915V1</div>
Crystal/Ceramic resonators	<div></div> <div>MSv31916V1</div>

External crystal/ceramic resonator (HSE crystal)

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

The associated hardware configuration is shown in [Figure 37](#). Refer to the electrical characteristics section of the *datasheet* for more details.

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

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

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

External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 32 MHz. You select this mode by setting the HSEBYP and HSEON bits in the [Clock control register \(RCC_CR\)](#). The external clock signal (square, sinus or triangle) with ~40-60% duty cycle depending on the frequency (refer to the *datasheet*) has to drive the OSC_IN pin while the OSC_OUT pin can be used a GPIO. See [Figure 37](#).

10.2.2 HSI clock

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

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

Calibration

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

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

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

For more details on how to measure the HSI frequency variation, refer to [Section 10.2.14: Internal/external clock measurement with TIM16](#).

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

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

The HSI signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 10.2.7: Clock security system (CSS) on page 182*.

10.2.3 PLL

The internal PLL can be used to multiply the HSI or HSE output clock frequency. Refer to *Figure 34* and *Clock control register (RCC_CR)*.

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

To modify the PLL configuration, proceed as follows:

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

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

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

10.2.4 LSE clock

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

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

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

Caution: To switch ON the LSE oscillator, 4096 LSE clock pulses need to be seen by an internal stabilization counter after the LSEON bit is set. Even in the case that no crystal or resonator is connected to the device, excessive external noise on the OSC32_IN pin may still lead the oscillator to start. Once the oscillator is started, it needs another 6 LSE clock pulses to complete a switching OFF sequence. If for any reason the oscillations are no more present on the OSC_IN pin, the oscillator cannot be switched OFF, locking the OSC32 pins from any other use and introducing unwanted power consumption. The only way to recover such situation is to perform the RTC domain reset by software.

External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. You select this mode by setting the LSEBYP and LSEON bits in the *RTC domain control register (RCC_BDCR)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin can be used as GPIO. See *Figure 37*.

10.2.5 LSI clock

The LSI RC acts as an low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and RTC. The clock frequency is around 40 kHz (between 30 kHz and 50 kHz). For more details, refer to the electrical characteristics section of the datasheets.

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

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

10.2.6 System clock (SYSCLK) selection

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

- HSI oscillator
- HSE oscillator
- PLL

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

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the switch will occur when the clock source becomes ready. Status bits in the *Clock control register (RCC_CR)* indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

10.2.7 Clock security system (CSS)

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

If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled, a clock failure event is sent to the break input of the advanced-control timers (TIM1 and TIM15/16/17) and an interrupt is generated to inform the software about the failure (Clock Security System Interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the Cortex-M4® F NMI (Non-Maskable Interrupt) exception vector.

Note: Once the CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and an NMI is automatically generated. The NMI will be executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the *Clock interrupt register (RCC_CIR)*.

If the HSE oscillator is used directly or indirectly as the system clock (indirectly means: it is used as PLL input clock, and the PLL clock is used as system clock), a detected failure

causes a switch of the system clock to the HSI oscillator and the disabling of the HSE oscillator. If the HSE clock (divided or not) is the clock entry of the PLL used as system clock when the failure occurs, the PLL is disabled too.

10.2.8 ADC clock

The ADC clock is derived from the PLL output. It can reach 72 MHz and can be divided by the following prescalers values: 1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128 or 256. It is asynchronous to the AHB clock. Alternatively, the ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). This programmable factor is configured using the CKMODE bit fields in the ADCx_CCR.

If the programmed factor is '1', the AHB prescaler must be set to '1'.

10.2.9 RTC clock

The RTCCLK clock source can be either the HSE/32, LSE or LSI clock. It is selected by programming the RTCSEL[1:0] bits in the *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 32 is used as the RTC clock:
 - The RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the 1.8 V domain).

10.2.10 Timers (TIMx) clock

APB clock source

The timers clock frequencies are automatically defined by hardware. There are two cases:

1. If the APB prescaler equals 1, the timer clock frequencies are set to the same frequency as that of the APB domain.
2. Otherwise, they are set to twice ($\times 2$) the frequency of the APB domain.

PLL clock source

A clock issued from the PLL (PLLCLKx2) can be selected for TIMx ($x = 1$ on the STM32F302xB/C; $x = 1, 15, 16$ and 17 on the STM32F302x6/8). This configuration allows to feed TIMx with a frequency up to 144 MHz when the system clock source is the PLL.

In this configuration:

- On the STM32F302xB/C, AHB and APB2 prescalers are set to 1, i.e. AHB and APB2 clocks are not divided with respect to the system clock.
- On the STM32F302x6/8 AHB or APB2 subsystem clocks are not divided by more than 2 cumulatively with respect to the system clock.

10.2.11 Watchdog clock

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

10.2.12 I2S clock

The I2S clock can be either the System clock or an external clock provided on I2S_CKIN pin. The selection of the I2S clock source is performed using bit 23 (I2SSRC) of RCC_CFGR register.

10.2.13 Clock-out capability

The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin. The configuration registers of the corresponding GPIO port must be programmed in alternate function mode. One of 5 clock signals can be selected as the MCO clock.

- LSI
- LSE
- SYSCLK
- HSI
- HSE
- PLL clock divided by 2 on the STM32F302xBxC and PLL clock not divided or divided by 2 on the STM32F302x6/8 and STM32F302xD/E, using the PLLNODIV bit in RCC_CFGR register

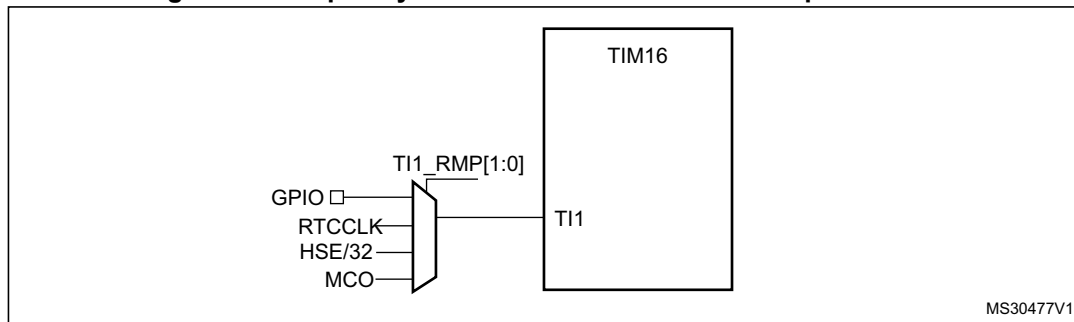
The selection is controlled by the MCO[2:0] bits in the *Clock configuration register (RCC_CFGR)*.

On the STM32F302x6/8 and STM32F302xD/E, the MCO frequency can be reduced by a configurable divider, controlled by the MCOPRE[2..0] bits of the Clock configuration register (RCC_CFGR).

10.2.14 Internal/external clock measurement with TIM16

It is possible to indirectly measure the frequency of all on-board clock sources by mean of the TIM16 channel 1 input capture. As represented on [Figure 38](#).

Figure 38. Frequency measurement with TIM16 in capture mode



The input capture channel of the Timer 16 can be a GPIO line or an internal clock of the MCU. This selection is performed through the T11_RMP [1:0] bits in the TIM16_OR register. The possibilities available are the following ones.

- TIM16 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.
- TIM16 Channel1 is connected to the RTCCLK.
- TIM16 Channel1 is connected to the HSE/32 Clock.
- TIM16 Channel1 is connected to the microcontroller clock output (MCO), this selection is controlled by the MCO[2:0] bits of the Clock configuration register (RCC_CFGR).

Calibration of the HSI

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

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

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

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

Calibration of the LSI

The calibration of the LSI will follow the same pattern that for the HSI, but changing the reference clock. It will be necessary to connect LSI clock to the channel 1 input capture of the TIM16. Then define the HSE as system clock source, the number of his clock counts between consecutive edges of the LSI signal provides a measure of the internal low speed clock period.

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

10.3 Low-power modes

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

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

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

All U(S)ARTs and I2Cs have the capability to enable the HSI oscillator even when the MCU is in Stop mode (if HSI is selected as the clock source for that peripheral).

All U(S)ARTs can also be driven by the LSE oscillator when the system is in Stop mode (if LSE is selected as clock source for that peripheral) and the LSE oscillator is enabled (LSEON) but they do not have the capability to turn on the LSE oscillator.

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

The CPU's deepsleep mode can be overridden for debugging by setting the DBG_STOP or DBG_STANDBY bits in the DBGMCU_CR register.

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

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

10.4 RCC registers

Refer to [Section 2.1 on page 42](#) for a list of abbreviations used in register descriptions.

10.4.1 Clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	PLL RDY	PLLON	Res	Res	Res	Res	CSS ON	HSE BYP	HSE RDY	HSE ON
						r	rw					rw	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

Bits 31:26 Reserved, must be kept at reset value.

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

Set by hardware to indicate that the PLL is locked.

0: PLL unlocked

1: PLL locked

Bit 24 **PLLON**: PLL enable

Set and cleared by software to enable PLL.

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

0: PLL OFF

1: PLL ON

Bits 23:20 Reserved, must be kept at reset value.

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

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

0: Clock detector OFF

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

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

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

0: HSE crystal oscillator not bypassed

1: HSE crystal oscillator bypassed with external clock

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

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

0: HSE oscillator not ready

1: HSE oscillator ready

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering Stop or Standby mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.

0: HSE oscillator OFF

1: HSE oscillator ON

Bits 15:8 **HSICAL[7:0]**: HSI clock calibration

These bits are initialized automatically at startup.

Bits 7:3 **HSITRIM[4:0]**: HSI clock trimming

These bits provide an additional user-programmable trimming value that is added to the HSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI.

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

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HSIRDY**: HSI clock ready flag

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

0: HSI oscillator not ready

1: HSI oscillator ready

Bit 0 **HSION**: HSI clock enable

Set and cleared by software.

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

0: HSI oscillator OFF

1: HSI oscillator ON

10.4.2 Clock configuration register (RCC_CFGR)

Address offset: 0x04

Reset value: 0x0000 0000

Access: $0 \leq \text{wait state} \leq 2$, word, half-word and byte access

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLNO DIV	MCOPRE[2:1]		MCOF/ MCOP RE0	Res	MCO[2:0]			I2SSSRC	USBPR ES	PLLMUL[3:0]				PLL XTPRE	PLL SRC
rw	rw	rw	r / 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
PLLSR C ⁽¹⁾	Res	PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

1. STM32F302xD/E only

Bit 31 **PLLNODIV**: Do not divide PLL to MCO (in STM32F302x6/8 and STM32F302xD/E only)

This bit is set and cleared by software. It switch-off divider-by-2 for PLL connection to MCO

0: PLL is divided by 2 before MCO

1: PLL is not divided before MCO

Note: In the STM32F302xB/C, this bit is reserved and must be kept at reset value.

Bits 30:28 **MCOPRE**: Microcontroller Clock Output Prescaler (in STM32F302x6/8 and STM32F302xD/E only)

These bits are set and cleared by software. It is highly recommended to change this prescaler before MCO output is enabled

000: MCO is divided by 1

001: MCO is divided by 2

010: MCO is divided by 4

.....

111: MCO is divided by 128

Note: In the STM32F302xB/C, bits 29 and 30 are reserved and must be kept at reset value.

Bit 28 **MCOF**: Microcontroller Clock Output Flag (STM32F302xB/C only)

Set and reset by hardware.

It is reset by hardware when MCO field is written with a new value

It is set by hardware when the switch to the new MCO source is effective.

Note: In the STM32F302x6/8/D/E, this bit is used with Bits 29 and 30 to select the Microcontroller Clock Output Prescaler.

Bit 27 Reserved, must be kept at reset value.

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

Set and cleared by software.

000: MCO output disabled, no clock on MCO

001: Reserved

010: LSI clock selected.

011: LSE clock selected.

100: System clock (SYSCLK) selected

101: HSI clock selected

110: HSE clock selected

111: PLL clock selected (divided by 1 or 2 depending on PLLNODIV bit).

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

Bit 23 **I2SSRC**: I2S external clock source selection

Set and reset by software to clock I2S2 and I2S3 with an external clock. This bits must be valid before enabling I2S2-3 clocks.

0: I2S2 and I2S3 clocked by system clock

1: I2S2 and I2S3 clocked by the external clock

Bit 22 **USBPRE**: USB prescaler

Set and reset by software to generate 48 MHz USB clock. These bits must be valid before enabling USB clocks.

0: PLL clock is divided by 1,5

1: PLL clock is not divided

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

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

Caution: The PLL output frequency must not exceed 72 MHz.

0000: PLL input clock x 2

0001: PLL input clock x 3

0010: PLL input clock x 4

0011: PLL input clock x 5

0100: PLL input clock x 6

0101: PLL input clock x 7

0110: PLL input clock x 8

0111: PLL input clock x 9

1000: PLL input clock x 10

1001: PLL input clock x 11

1010: PLL input clock x 12

1011: PLL input clock x 13

1100: PLL input clock x 14

1101: PLL input clock x 15

1110: PLL input clock x 16

1111: PLL input clock x 16

Bit 17 **PLLXTPRE**: HSE divider for PLL input clock

This bits is set and cleared by software to select the HSE division factor for the PLL. It can be written only when the PLL is disabled.

Note: This bit is the same as the LSB of PREDIV in [Clock configuration register 2 \(RCC_CFGR2\)](#) (for compatibility with other STM32 products)

0000: HSE input to PLL not divided

0001: HSE input to PLL divided by 2

- Bits 16:15 **PLL SRC**: PLL entry clock source (STM32F302xD/E only)
Set and cleared by software to select PLL clock source. These bits can be written only when PLL is disabled.
- 00: HSI/2 used as PREDIV1 entry and PREDIV1 forced to div by 2.
 - 01: HSI used as PREDIV1 entry.
 - 10: HSE used as PREDIV1 entry.
 - 11: Reserved.
- Bit 16 **PLL SRC**: PLL entry clock source
Set and cleared by software to select PLL clock source. This bit can be written only when PLL is disabled.
- 0: HSI/2 selected as PLL input clock
 - 1: HSE/PREDIV selected as PLL input clock (refer to [Section 10.4.12: Clock configuration register 2 \(RCC_CFGR2\)](#) on page 209)
- Bit 15 Reserved, must be kept at reset value in STM32F302xB/C and STM32F302x6/8 devices, and used with Bit 16 in STM32F302xD/E to select the PLL clock source.
- Bit 14 Reserved, must be kept at reset value.
- Bits 13:11 **PPRE2**: APB high-speed prescaler (APB2)
Set and cleared by software to control the division factor of the APB2 clock (PCLK).
- 0xx: HCLK not divided
 - 100: HCLK divided by 2
 - 101: HCLK divided by 4
 - 110: HCLK divided by 8
 - 111: HCLK divided by 16
- Bits 10:8 **PPRE1**: APB Low-speed prescaler (APB1)
Set and cleared by software to control the division factor of the APB1 clock (PCLK).
- 0xx: HCLK not divided
 - 100: HCLK divided by 2
 - 101: HCLK divided by 4
 - 110: HCLK divided by 8
 - 111: HCLK divided by 16

Bits 7:4 **HPRE**: HCLK prescaler

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

0xxx: SYSCLK not divided
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512

Note: The prefetch buffer must be kept on when using a prescaler different from 1 on the AHB clock. Refer to section [Read operations on page 59](#) for more details.

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

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

00: HSI oscillator used as system clock
01: HSE oscillator used as system clock
10: PLL used as system clock
11: not applicable

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

Set and cleared by software to select SYSCLK source.

Cleared by hardware to force HSI selection when leaving Stop and Standby mode or in case of failure of the HSE oscillator used directly or indirectly as system clock (if the Clock Security System is enabled).

00: HSI selected as system clock
01: HSE selected as system clock
10: PLL selected as system clock
11: not allowed

10.4.3 Clock interrupt register (RCC_CIR)

Address offset: 0x08

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	CSSC	Res	Res	PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								w			w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	Res	Res	PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
			rw	rw	rw	rw	rw	r			r	r	r	r	r

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

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

This bit is set by software to clear the CSSF flag.

0: No effect

1: Clear CSSF flag

Bits 22:21 Reserved, must be kept at reset value.

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

This bit is set by software to clear the PLLRDYF flag.

0: No effect

1: Clear PLLRDYF flag

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

This bit is set by software to clear the HSERDYF flag.

0: No effect

1: Clear HSERDYF flag

Bit 18 **HSIRDYC**: HSI ready interrupt clear

This bit is set software to clear the HSIRDYF flag.

0: No effect

1: Clear HSIRDYF flag

Bit 17 **LSERDYC**: LSE ready interrupt clear

This bit is set by software to clear the LSERDYF flag.

0: No effect

1: LSERDYF cleared

Bit 16 **LSIRDYC**: LSI ready interrupt clear

This bit is set by software to clear the LSIRDYF flag.

0: No effect

1: LSIRDYF cleared

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PLL RDYIE**: PLL ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by PLL lock.

0: PLL lock interrupt disabled

1: PLL lock interrupt enabled

- Bit 11 **HSERDYIE**: HSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled
1: HSE ready interrupt enabled
- Bit 10 **HSIRDYIE**: HSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled
1: HSI ready interrupt enabled
- Bit 9 **LSERDYIE**: LSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled
1: LSE ready interrupt enabled
- Bit 8 **LSIRDYIE**: LSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the LSI oscillator stabilization.
0: LSI ready interrupt disabled
1: LSI ready interrupt enabled
- Bit 7 **CSSF**: Clock security system interrupt flag
Set by hardware when a failure is detected in the HSE oscillator.
Cleared by software setting the CSSC bit.
0: No clock security interrupt caused by HSE clock failure
1: Clock security interrupt caused by HSE clock failure
- Bits 6:5 Reserved, must be kept at reset value.
- Bit 4 **PLLRDYF**: PLL ready interrupt flag
Set by hardware when the PLL locks and PLLRDYDIE is set.
Cleared by software setting the PLLRDYC bit.
0: No clock ready interrupt caused by PLL lock
1: Clock ready interrupt caused by PLL lock
- Bit 3 **HSERDYF**: HSE ready interrupt flag
Set by hardware when the HSE clock becomes stable and HSERDYDIE is set.
Cleared by software setting the HSERDYC bit.
0: No clock ready interrupt caused by the HSE oscillator
1: Clock ready interrupt caused by the HSE oscillator

Bit 2 HSIRDYF: HSI ready interrupt flag

Set by hardware when the HSI clock becomes stable and HSIRDYDIE is set in a response to setting the HSION (refer to [Clock control register \(RCC_CR\)](#)). When HSION is not set but the HSI oscillator is enabled by the peripheral through a clock request, this bit is not set and no interrupt is generated.

Cleared by software setting the HSIRDYC bit.

0: No clock ready interrupt caused by the HSI oscillator

1: Clock ready interrupt caused by the HSI oscillator

Bit 1 LSERDYF: LSE ready interrupt flag

Set by hardware when the LSE clock becomes stable and LSERDYDIE is set.

Cleared by software setting the LSERDYC bit.

0: No clock ready interrupt caused by the LSE oscillator

1: Clock ready interrupt caused by the LSE oscillator

Bit 0 LSIRDYF: LSI ready interrupt flag

Set by hardware when the LSI clock becomes stable and LSIRDYDIE is set.

Cleared by software setting the LSIRDYC bit.

0: No clock ready interrupt caused by the LSI oscillator

1: Clock ready interrupt caused by the LSI oscillator

10.4.4 APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x0C

Reset value: 0x00000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TIM17 RST	TIM16 RST	TIM15 RST
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI4RST	USART1 RST	Res	SPI1 RST ⁽¹⁾	TIM1 RST	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SYS CFG RST
rw	rw		rw	rw											rw

1. Available only on STM32F302xB/C/D/E devices.

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

Bit 18 TIM17RST: TIM17 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM17 timer

Bit 17 TIM16RST: TIM16 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM16 timer

- Bit 16 **TIM15RST**: TIM15 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM15 timer
- Bit 15 **SPI4RST**: SPI4 reset (only on STM32F302xD/E devices)
Set and cleared by software.
0: No effect
1: Reset SPI4
- 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 (STM32F302xB/C/D/E devices only)
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, Comparators and operational amplifiers reset
Set and cleared by software.
0: No effect
1: Reset SYSCFG, COMP, and OPAMP

10.4.5 APB1 peripheral reset register (RCC_APB1RSTR)

Address offset: 0x10

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	I2C3 RST	DAC1 RST	PWR RST	Res	Res	CAN RST	Res	USB RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	USART3 RST	USART2 RST	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
SPI3 RST	SPI2 RST	Res	Res	WWDG RST	Res	Res	Res	Res	Res	Res	TIM6 RST	Res	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw							rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **I2C3RST**: I2C3 reset

Set and cleared by software.

0: No effect

1: Reset I2C3

Bit 29 **DAC1RST**: DAC1 interface reset

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 power interface

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **CANRST**: CAN reset

Set and reset by software.

0: does not reset the CAN

1: resets the CAN

Bit 24 Reserved, must be kept at reset value

Bit 23 **USBRST**: USB reset

Set and reset by software.

0: does not reset USB

1: resets USB

Bit 22 **I2C2RST**: I2C2 reset

Set and cleared by software.

0: No effect

1: Reset I2C2

Bit 21 **I2C1RST**: I2C1 reset

Set and cleared by software.

0: No effect

1: Reset I2C1

- Bit 20 **UART5RST**: UART5 reset (STM32F302xB/C/D/E devices only)
Set and cleared by software.
0: No effect
1: Reset UART5
- Bit 19 **UART4RST**: UART4 reset (STM32F302xB/C/D/E devices only)
Set and cleared by software.
0: No effect
1: Reset UART4
- Bit 18 **USART3RST**: USART3 reset
Set and cleared by software.
0: No effect
1: Reset USART3
- Bit 17 **USART2RST**: USART2 reset
Set and cleared by software.
0: No effect
1: Reset USART2
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **SPI3RST**: SPI3 reset
Set and cleared by software.
0: No effect
1: Reset SPI3 and I2S3
- Bit 14 **SPI2RST**: SPI2 reset
Set and cleared by software.
0: No effect
1: Reset SPI2 and I2S2
- Bits 13:12 Reserved, must be kept at reset value.
- Bit 11 **WWDGRST**: Window watchdog reset
Set and cleared by software.
0: No effect
1: Reset window watchdog
- Bits 10:5 Reserved, must be kept at reset value.
- Bit 4 **TIM6RST**: TIM6 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM6
- Bit 3 Reserved, must be kept at reset value.

Bit 2 **TIM4RST**: TIM4 timer reset (STM32F302xB/C devices only)

Set and cleared by software.

0: No effect
1: Reset TIM4

Bit 1 **TIM3RST**: TIM3 timer reset (STM32F302xB/C devices only)

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

10.4.6 AHB peripheral clock enable register (RCC_AHBENR)

Address offset: 0x14

Reset value: 0x0000 0014

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	ADC12EN	Res	Res	Res	TSCEN	IOPGEN ⁽¹⁾	IOPFEN	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	IOPHEN ⁽¹⁾
			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	CRCEN	FMCEN ⁽¹⁾	FLITFEN	Res	SRAMEN	DMA2EN	DMA1EN
									rw	rw	rw		rw	rw	rw

1. Only on STM32F302xDxE.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADC12EN**: ADC1 and ADC2 enable (ADC2 only in STM32F302xB/C)

Set and reset by software.

0: ADC1 and ADC2 clock disabled
1: ADC1 and ADC2 clock enabled

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **TSCEN**: Touch sensing controller clock enable

Set and cleared by software.

0: TSC clock disabled
1: TSC clock enabled

Bit 23 **IOPGEN**: IO port G clock enable. (Only on STM32F302xDxE)

Set and cleared by software.

0: IO port G clock disabled
1: IO port G clock enabled

- Bit 22 **IOPFEN**: I/O port F clock enable
Set and cleared by software.
0: I/O port F clock disabled
1: I/O port F clock enabled
- Bit 21 **IOPEEN**: I/O port E clock enable(STM32F302xB/C devices only)
Set and cleared by software.
0: I/O port E clock disabled
1: I/O port E clock enabled.
- Bit 20 **IOPDEN**: I/O port D clock enable
Set and cleared by software.
0: I/O port D clock disabled
1: I/O port D clock enabled
- Bit 19 **IOPCEN**: I/O port C clock enable
Set and cleared by software.
0: I/O port C clock disabled
1: I/O port C clock enabled
- Bit 18 **IOPBEN**: I/O port B clock enable
Set and cleared by software.
0: I/O port B clock disabled
1: I/O port B clock enabled
- Bit 17 **IOPAEN**: I/O port A clock enable
Set and cleared by software.
0: I/O port A clock disabled
1: I/O port A clock enabled
- Bit 16 **IOPHEN**: IO port H clock enable. (Only on STM32F302xDxE)
Set and cleared by software.
0: IO port H clock disabled
1: IO port H clock enabled
- Bits 15:7 Reserved, must be kept at reset value.
- Bit 6 **CRCEN**: CRC clock enable
Set and cleared by software.
0: CRC clock disabled
1: CRC clock enabled
- Bit 5 **FMCEN**: FMC clock enable. (Only on STM32F302xDxE)
Set and cleared by software.
0: FMC clock disabled
1: FMC clock enabled
- Bit 4 **FLITFEN**: FLITF clock enable
Set and cleared by software to disable/enable FLITF clock during Sleep mode.
0: FLITF clock disabled during Sleep mode
1: FLITF clock enabled during Sleep mode
- Bit 3 Reserved, must be kept at reset value.

Bit 2 **SRAMEN**: SRAM interface clock enable

Set and cleared by software to disable/enable SRAM interface clock during Sleep mode.

0: SRAM interface clock disabled during Sleep mode.

1: SRAM interface clock enabled during Sleep mode

Bit 1 **DMA2EN**: DMA2 clock enable (STM32F302xB/C devices only)

Set and cleared by software.

0: DMA2 clock disabled

1: DMA2 clock enabled

Bit 0 **DMA1EN**: DMA1 clock enable

Set and cleared by software.

0: DMA1 clock disabled

1: DMA1 clock enabled

10.4.7 APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x18

Reset value: 0x0000 0000

Access: word, half-word and byte access

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TIM17 EN	TIM16 EN	TIM15 EN
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI4EN	USART1EN	Res	SPI1 EN	TIM1 EN	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SYS CFGEN
rw	rw		rw	rw											rw

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

Bit 18 **TIM17EN**: TIM17 timer clock enable

Set and cleared by software.

0: TIM17 timer clock disabled

1: TIM17 timer clock enabled

Bit 17 **TIM16EN**: TIM16 timer clock enable

Set and cleared by software.

0: TIM16 timer clock disabled

1: TIM16 timer clock enabled

Bit 16 **TIM15EN**: TIM15 timer clock enable

Set and cleared by software.

0: TIM15 timer clock disabled

1: TIM15 timer clock enabled

Bit 15 **SPI4EN**: SPI4 clock enable (STM32F302xD/E only)

Set and cleared by software.

0: SPI4 clock disabled

1: SPI4 clock enabled

Bit 14 **USART1EN**: USART1 clock enable

Set and cleared by software.

0: USART1 clock disabled

1: USART1 clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN**: SPI1 clock enable (STM32F302xB/C devices only)

Set and cleared by software.

0: SPI1 clock disabled

1: SPI1 clock enabled

Bit 11 **TIM1EN**: TIM1 timer clock enable

Set and cleared by software.

0: TIM1 timer clock disabled

1: TIM1 timer clock enabled

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGEN**: SYSCFG clock enable

Set and cleared by software.

0: SYSCFG clock disabled

1: SYSCFG clock enabled

10.4.8 APB1 peripheral clock enable register (RCC_APB1ENR)

Address: 0x1C

Reset value: 0x0000 0000

Access: word, half-word and byte access

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	I2C3 EN	DAC1 EN	PWR EN	Res	Res	CAN EN	Res	USB EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	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
SPI3 EN	SPI2 EN	Res	Res	WWD GEN	Res	Res	Res	Res	Res	Res	TIM6EN	Res	TIM4EN	TIM3EN	TIM2 EN
rw	rw			rw							rw		rw	rw	rw

- Bit 31 Reserved, must be kept at reset value.
- Bit 30 **I2C3EN**: I2C3 clock enable (only in STM32F302x6/8 and STM32F302xD/E devices)
Set and cleared by software.
0: I2C3 clock disabled
1: I2C3 clock enabled
- Bit 29 **DAC1EN**: DAC1 interface clock enable
Set and cleared by software.
0: DAC1 interface clock disabled
1: DAC1 interface clock enabled
- Bit 28 **PWREN**: Power interface clock enable
Set and cleared by software.
0: Power interface clock disabled
1: Power interface clock enabled
- Bits 27:263 Reserved, must be kept at reset value.
- Bit 25 **CANEN**: CAN clock enable
Set and reset by software.
0: CAN clock disabled
1: CAN clock enabled
- Bit 24 Reserved, must be kept at reset value.
- Bit 23 **USBEN**: USB clock enable
Set and reset by software.
0: USB clock disabled
1: USB clock enabled
- Bit 22 **I2C2EN**: I2C2 clock enable
Set and cleared by software.
0: I2C2 clock disabled
1: I2C2 clock enabled
- Bit 21 **I2C1EN**: I2C1 clock enable
Set and cleared by software.
0: I2C1 clock disabled
1: I2C1 clock enabled
- Bit 20 **UART5EN**: UART5 clock enable (STM32F302xB/C/D/E devices only)
Set and cleared by software.
0: UART5 clock disabled
1: UART5 clock enabled
- Bit 19 **UART4EN**: UART4 clock enable (STM32F302xB/C/D/E devices only)
Set and cleared by software.
0: UART4 clock disabled
1: UART4 clock enabled
- Bit 18 **USART3EN**: USART3 clock enable
Set and cleared by software.
0: USART3 clock disabled
1: USART3 clock enabled

- Bit 17 **USART2EN**: USART2 clock enable
Set and cleared by software.
0: USART2 clock disabled
1: USART2 clock enabled
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **SPI3EN**: SPI3 clock enable
Set and cleared by software.
0: SPI3 clock disabled
1: SPI3 clock enabled
- Bit 14 **SPI2EN**: SPI2 clock enable
Set and cleared by software.
0: SPI2 clock disabled
1: SPI2 clock enabled
- Bits 13:12 Reserved, must be kept at reset value.
- Bit 11 **WWDGEN**: Window watchdog clock enable
Set and cleared by software.
0: Window watchdog clock disabled
1: Window watchdog clock enabled
- Bits 10:5 Reserved, must be kept at reset value.
- Bit 4 **TIM6EN**: TIM6 timer clock enable
Set and cleared by software.
0: TIM6 clock disabled
1: TIM6 clock enabled
- Bit 3 Reserved, must be kept at reset value.
- Bit 2 **TIM4EN**: TIM4 timer clock enable (STM32F302xB/C/D/E devices only)
Set and cleared by software.
0: TIM4 clock disabled
1: TIM4 clock enabled
- Bit 1 **TIM3EN**: TIM3 timer clock enable (STM32F302xB/C devices only)
Set and cleared by software.
0: TIM3 clock disabled
1: TIM3 clock enabled
- Bit 0 **TIM2EN**: TIM2 timer clock enable
Set and cleared by software.
0: TIM2 clock disabled
1: TIM2 clock enabled

10.4.9 RTC domain control register (RCC_BDCR)

Address offset: 0x20

Reset value: 0x0000 0018h reset by RTC domain Reset.

Access: $0 \leq \text{wait state} \leq 3$, word, half-word and byte access

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

Note: The *LSEON*, *LSEBYP*, *RTCSEL* and *RTCEN* bits of the *RTC domain control register (RCC_BDCR)* are in the RTC domain. As a result, after Reset, these bits are write-protected and the *DBP* bit in the *Power control register (PWR_CR)* has to be set before these can be modified. These bits are only reset after a RTC domain Reset (see [Section 10.1.3: RTC domain reset](#)). Any internal or external Reset will not have any effect on these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BDRST
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	Res	Res	Res	Res	Res	RTCSEL[1:0]		Res	Res	Res	LSEDRV[1:0]		LSE BYP	LSE RDY	LSEON
rw						rw	rw				rw	rw	rw	r	rw

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

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

Set and cleared by software.

0: Reset not activated

1: Resets the entire RTC domain

Bit 15 **RTCEN**: RTC clock enable

Set and cleared by software.

0: RTC clock disabled

1: RTC clock enabled

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

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

Set by software to select the clock source for the RTC. Once the RTC clock source has been selected, it cannot be changed anymore unless the RTC domain is reset. The BDRST bit can be used to reset them.

00: No clock

01: LSE oscillator clock used as RTC clock

10: LSI oscillator clock used as RTC clock

11: HSE oscillator clock divided by 32 used as RTC clock

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

Bits 4:3 **LSEDRV** LSE oscillator drive capability

Set and reset by software to modulate the LSE oscillator's drive capability. A reset of the RTC domain restores the default value.

00: 'Xtal mode' lower driving capability

01: 'Xtal mode' medium low driving capability

10: 'Xtal mode' medium high driving capability

11: 'Xtal mode' higher driving capability (reset value)

Note: The oscillator is in Xtal mode when it is not in bypass mode.

Bit 2 **LSEBYP**: LSE oscillator bypass

Set and cleared by software to bypass oscillator in debug mode. This bit can be written only when the external 32 kHz oscillator is disabled.

0: LSE oscillator not bypassed

1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready

Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles.

0: LSE oscillator not ready

1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enable

Set and cleared by software.

0: LSE oscillator OFF

1: LSE oscillator ON

10.4.10 Control/status register (RCC_CSR)

Address: 0x24

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

Access: $0 \leq \text{wait state} \leq 3$, word, half-word and byte access

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	RMVF	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	LSI RDY	LSION
														r	rw

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

Bit 24 **RMVF**: Remove reset flag

Set by software to clear the reset flags.

0: No effect

1: Clear the reset flags

Bits 23:2 Reserved, must be kept at reset value.

Bit 1 **LSIRDY**: LSI oscillator ready

Set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after 3 LSI oscillator clock cycles.

0: LSI oscillator not ready

1: LSI oscillator ready

Bit 0 **LSION**: LSI oscillator enable

Set and cleared by software.

0: LSI oscillator OFF

1: LSI oscillator ON

10.4.11 AHB peripheral reset register (RCC_AHBRSTR)

Address: 0x28

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	ADC12 RST	Res	Res	Res	TSC RST	IOPGR ST ⁽¹⁾	IOPF RST	IOPE RST	IOPD RST	IOPC RST	IOPB RST	IOPA RST	IOPHR ST ⁽¹⁾
			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	FMC RST	Res	Res	Res	Res	Res

1. Only on STM32F302xDxE.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADC12RST**: ADC1 and ADC2 reset (only ADC1 on STM32F302x6/8 devices)

Set and reset by software.

0: does not reset the ADC1 and ADC2

1: resets the ADC1 and ADC2

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **TSCRST**: Touch sensing controller reset

Set and cleared by software.

0: No effect

1: Reset TSC

Bit 23 **IOPGRST**: I/O port G reset. (Only on STM32F302xDxE)

Set and cleared by software.

0: No effect

1: Reset I/O port G

Bit 22 **IOPFRST**: I/O port F reset

Set and cleared by software.

0: No effect

1: Reset I/O port F

Bit 21 **OPERST**: I/O port E reset (STM32F302xB/C devices only)

Set and cleared by software.

0: No effect

1: Reset I/O port E

Bit 20 **IOPDRST**: I/O port D reset

Set and cleared by software.

0: No effect

1: Reset I/O port D

Bit 19 **IOPCRST**: I/O port C reset

Set and cleared by software.

0: No effect

1: Reset I/O port C

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

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

Bit 16 **IOPHRST**: I/O port H reset (Only on STM32F302xDxE).
Set and cleared by software.
0: No effect
1: Reset I/O port H

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **FMCRST**: FMC reset (Only on STM32F302xDxE).
Set and cleared by software.
0: No effect
1: Reset FMC

Bits 4:0 Reserved, must be kept at reset value.

10.4.12 Clock configuration register 2 (RCC_CFGR2)

Address: 0x2C

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	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	ADC12PRES[4:0]					PREDIV[3:0]			
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 8:4 **ADC12PRES**: ADC12 prescaler (ADC1 prescaler in STM32F302x6/8)

Set and reset by software to control PLL clock to ADC12 division factor.

0xxxx: ADC12 clock disabled, ADC12 can use AHB clock

10000: PLL clock divided by 1

10001: PLL clock divided by 2

10010: PLL clock divided by 4

10011: PLL clock divided by 6

10100: PLL clock divided by 8

10101: PLL clock divided by 10

10110: PLL clock divided by 12

10111: PLL clock divided by 16

11000: PLL clock divided by 32

11001: PLL clock divided by 64

11010: PLL clock divided by 128

11011: PLL clock divided by 256

others: PLL clock divided by 256

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

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

Note: Bit 0 is the same bit as bit17 in Clock configuration register (RCC_CFGR), so modifying bit17 Clock configuration register (RCC_CFGR) also modifies bit 0 in Clock configuration register 2 (RCC_CFGR2) (for compatibility with other STM32 products)

0000: HSE input to PLL not divided

0001: HSE input to PLL divided by 2

0010: HSE input to PLL divided by 3

0011: HSE input to PLL divided by 4

0100: HSE input to PLL divided by 5

0101: HSE input to PLL divided by 6

0110: HSE input to PLL divided by 7

0111: HSE input to PLL divided by 8

1000: HSE input to PLL divided by 9

1001: HSE input to PLL divided by 10

1010: HSE input to PLL divided by 11

1011: HSE input to PLL divided by 12

1100: HSE input to PLL divided by 13

1101: HSE input to PLL divided by 14

1110: HSE input to PLL divided by 15

1111: HSE input to PLL divided by 16

10.4.13 Clock configuration register 3 (RCC_CFGR3)

Address: 0x30

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	TIM34 SW ⁽¹⁾	TIM2 SW ⁽¹⁾	UART5SW[1:0]		UART4SW[1:0]		USART3SW[1:0]		USART2SW[1:0]	
						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	TIM17 SW	Res	TIM16 SW	TIM15 SW	Res	TIM1 SW	Res	I2C3 SW	I2C2 SW	I2C1 SW	Res	Res	USART1SW[1:0]	
		rw		rw	rw		rw		rw	rw	rw			rw	rw

1. Only on STM32F302xDxE.

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TIM34SW**: Timer34 clock source selection

Set and reset by software to select TIM34 clock source.

The bit is writable only when the following conditions occur: system clock source is the PLL and AHB or APB2 subsystem clocks are not divided by more than 2 cumulatively.

The bit is reset by hardware when exiting from the previous condition (user must set the bit again in case of a new switch is required)

0: PCLK2 clock (doubled frequency when prescaled) (default)

1: PLL vco output (running up to 144 MHz)

Note: STM32F302xDxE only.

Bit 24 **TIM2SW**: Timer2 clock source selection

Set and reset by software to select TIM2 clock source.

The bit is writable only when the following conditions occur: clock system = PLL, and AHB or APB2 subsystem clocks are not divided by more than 2 cumulatively.

0: PCLK2 clock (doubled frequency when prescaled) (default)

1: PLL vco output (running up to 144 MHz)

Note: STM32F302xDxE only.

Bits 23:22 **UART5SW[1:0]**: UART5 clock source selection (STM32F302xB/C devices only)

This bit is set and cleared by software to select the UART5 clock source.

00: PCLK selected as UART5 clock source (default)

01: System clock (SYSCLK) selected as UART5 clock

10: LSE clock selected as UART5 clock

11: HSI clock selected as UART5 clock

Bits 21:20 **UART4SW[1:0]**: UART4 clock source selection (STM32F302xB/C devices only)

This bit is set and cleared by software to select the UART4 clock source.

00: PCLK selected as UART4 clock source (default)

01: System clock (SYSCLK) selected as UART4 clock

10: LSE clock selected as UART4 clock

11: HSI clock selected as UART4 clock

- Bits 19:18 **USART3SW[1:0]**: USART3 clock source selection
This bit is set and cleared by software to select the USART3 clock source.
00: PCLK selected as USART3 clock source (default)
01: System clock (SYSCLK) selected as USART3 clock
10: LSE clock selected as USART3 clock
11: HSI clock selected as USART3 clock
- Bits 17:16 **USART2SW[1:0]**: USART2 clock source selection
This bit is set and cleared by software to select the USART2 clock source.
00: PCLK selected as USART2 clock source (default)
01: System clock (SYSCLK) selected as USART2 clock
10: LSE clock selected as USART2 clock
11: HSI clock selected as USART2 clock
- Bits 14:14 Reserved, must be kept at reset value.
- Bit 13 **TIM17SW**: Timer17 clock source selection
Set and reset by software to select TIM17 clock source.
The bit is writable only when the following conditions occur: system clock source is the PLL and AHB or APB2 subsystem clocks are not divided by more than 2 cumulatively.
The bit is reset by hardware when exiting from the previous condition (user must set the bit again in case of a new switch is required)
0: PCLK2 clock (doubled frequency when prescaled) (default)
1: PLL vco output (running up to 144 MHz)
Note: STM32F302x6/8 and STM32F302xD/E devices only.
- Bit 12 Reserved, must be kept at reset value.
- Bit 11 **TIM16SW**: Timer16 clock source selection
Set and reset by software to select TIM16 clock source.
The bit is writable only when the following conditions occur: system clock source is the PLL and AHB or APB2 subsystem clocks are not divided by more than 2 cumulatively.
The bit is reset by hardware when exiting from the previous condition (user must set the bit again in case of a new switch is required)
0: PCLK2 clock (doubled frequency when prescaled) (default)
1: PLL vco output (running up to 144 MHz)
Note: STM32F302x6/8 and STM32F302xD/E devices only.
- Bit 10 **TIM15SW**: Timer15 clock source selection
Set and reset by software to select TIM15 clock source.
The bit is writable only when the following conditions occur: system clock source is the PLL and AHB or APB2 subsystem clocks are not divided by more than 2 cumulatively.
The bit is reset by hardware when exiting from the previous condition (user must set the bit again in case of a new switch is required)
0: PCLK2 clock (doubled frequency when prescaled) (default)
1: PLL vco output (running up to 144 MHz)
Note: STM32F302x6/8 and STM32F302xD/E devices only.
- Bit 9 Reserved, must be kept at reset value.

Bit 8 TIM1SW: Timer1 clock source selection

Set and reset by software to select TIM1 clock source.

The bit is writable only when the following conditions occur: clock system = PLL, and AHB and APB2 subsystem clock not divided respect the clock system.

The bit is reset by hardware when exiting from the previous condition (user must set the bit again in case of a new switch is required)

0: PCLK2 clock (doubled frequency when prescaled) (default)

1: PLL vco output (running up to 144 MHz)

Bit 7 Reserved, must be kept at reset value.**Bit 6 I2C3SW:** I2C3 clock source selection (STM32F302x6/8 devices only)

This bit is set and cleared by software to select the I2C3 clock source.

0: HSI clock selected as I2C3 clock source (default)

1: SYSCLK clock selected as I2C3 clock

Bit 5 I2C2SW: I2C2 clock source selection

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

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

1: SYSCLK clock selected as I2C2 clock

Bit 4 I2C1SW: I2C1 clock source selection

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

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

1: SYSCLK clock selected as I2C1 clock

Bits 3:2 Reserved, must be kept at reset value.**Bits 1:0 USART1SW[1:0]:** USART1 clock source selection

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

00: PCLK selected as USART1 clock source (default)

01: System clock (SYSCLK) selected as USART1 clock

10: LSE clock selected as USART1 clock

11: HSI clock selected as USART1 clock

10.4.14 RCC register map

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

Table 66. RCC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	PLLRDY	PLLON	Res.	Res.	Res.	Res.	CSSON	HSEBYP	HSERDY	HSEON	HSICAL[7:0]							HSITRIM[4:0]				Res.	HSIRDY	HSION		
	Reset value							0	0					0	0	0	0	x	x	x	x	x	x	x	x	1	0	0	0	0		1	1
0x04	RCC_CFGR	PLLNO <div>(1)</div>	MCO <div>P</div> [2:1] ⁽¹⁾		MCO <div>P</div> [0] ⁽¹⁾ / MCO <div>F</div> [2]			MCO <div>P</div> [2:0]		I2SSRC	USBPRE	PLLMUL[3:0]				PLLXTPRE	PLLSRC	PLLSRC ⁽³⁾	Res.	PPRE2 <div>P</div> [2:0]			PPRE1 <div>P</div> [2:0]		HPRE[3:0]			SWS <div>P</div> [1:0]		SW <div>P</div> [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
0x08	RCC_CIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSC	Res.	Res.	Res.	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Res.	Res.	Res.	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Res.	Res.	Res.	Res.	Res.	
	Reset value									0				0	0	0	0	0				0	0	0	0	0	0			0	0	0	0
0x0C	RCC_APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM16RST	TIM15RST	SPI4RST ⁽³⁾	USART1RST	Res.	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0		0	0											SYSCFG
0x10	RCC_APB1RSTR	Res.	I2C3RST ⁽¹⁾	DAC1RST	PWRRST	Res.	Res.	CANRST	Res.	USBRST	I2C2RST	I2C1RST	UART5RST ⁽²⁾	UART4RST ⁽²⁾	USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	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
0x14	RCC_AHBENR	Res.	Res.	Res.	ADC12EN	Res.	Res.	Res.	TSCEN	Res.	IOPGEN ⁽³⁾	IOPFEN	IOPEEN ⁽²⁾	IOPDEN	IOPCEN	IOBPEN	IOPAEN	IOPHEN ⁽³⁾	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	1		1	0	0
0x18	RCC_APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	Res.	SPI4EN ⁽³⁾	USART1EN	Res.	SPI1EN ⁽²⁾	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0		0	0	0	0	0										SYSCFG
0x1C	RCC_APB1ENR	Res.	I2C3EN ⁽¹⁾	DAC1EN	PWREN	Res.	Res.	CANEN	Res.	USBEN	I2C2EN	I2C1EN	UART5EN ⁽²⁾	UART4EN ⁽²⁾	USART3EN	USART2EN	Res.	SP3EN	SP2EN	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

Table 66. RCC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	RCC_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC SEL [1:0]	Res.	Res.	Res.	Res.	Res.	LSE DRV [1:0]	Res.	LSEBYP	LSEON
	Reset value																0	0	Res.	Res.	Res.	Res.	Res.	0	0				1	1	0	0	0
0x24	RCC_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
	Reset value								0																							0	0
0x28	RCC_AHBRSTR	Res.	Res.	Res.	ADC12RST	Res.	Res.	Res.	TSCRST	IOPGRST ⁽³⁾	IOPFRST	IOPERST ⁽²⁾	IOPDRST	IOPCRST	IOPBRST	IOPARST	IOPHRST ⁽³⁾	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																	
0x2C	RCC_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12PRES [4:0]				PREDIV[3:0]					
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x30	RCC_CFGR3	Res.	Res.	Res.	Res.	Res.	Res.	TIM34SW ⁽³⁾	TIM2SW ⁽³⁾	UART5SW[1:0] ⁽²⁾	UART4SW[1:0] ⁽²⁾	USART3SW[1:0]	USART2SW[1:0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1SW[1:0]
	Reset value									0	0	0	0	0	0	0				0		0	0	0	0	0						0	0

1. On STM32F302xB/C devices only.

2. On STM32F302x6/8 devices only.

3. On STM32F302xD/E only

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

11 General-purpose I/Os (GPIO)

11.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

11.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the port A, B, C, D, E, F, G, H on STM32F302xD/E, A, B and D I/O configuration in STM32F302xB/C devices and port A, B, C, D and F in STM32F302x6/8 devices.
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every clock cycle
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

11.3 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 39 and Figure 40 show the basic structures of a standard and a 5 V tolerant I/O port bit, respectively. Table 67 gives the possible port bit configurations.

Figure 39. Basic structure of an I/O port bit

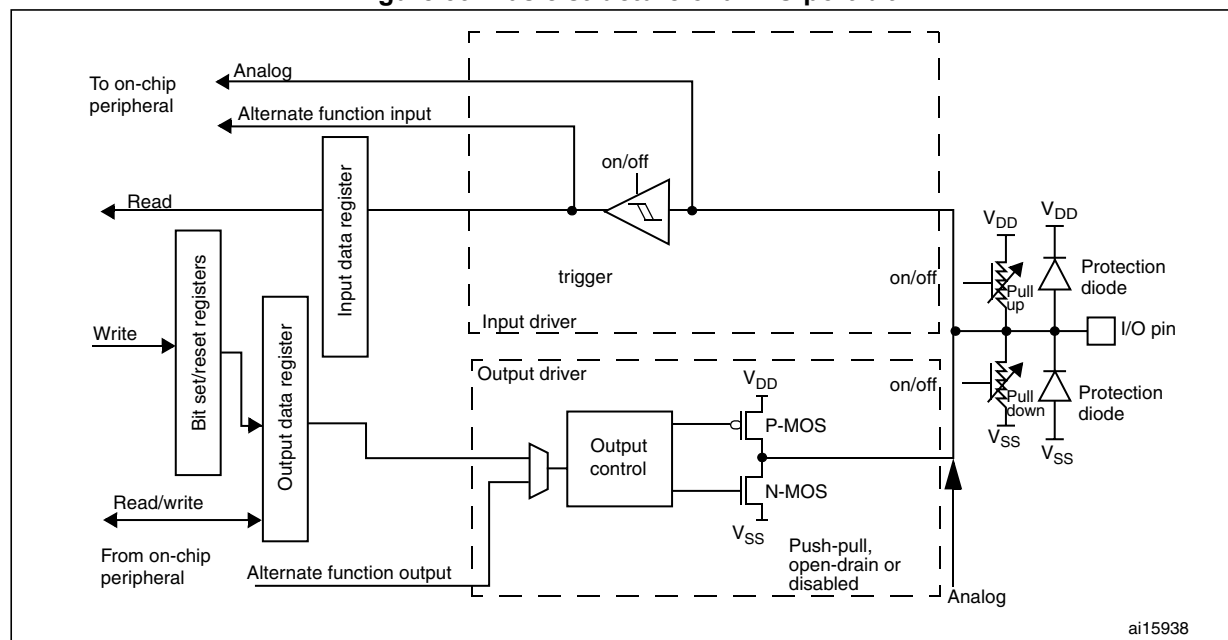
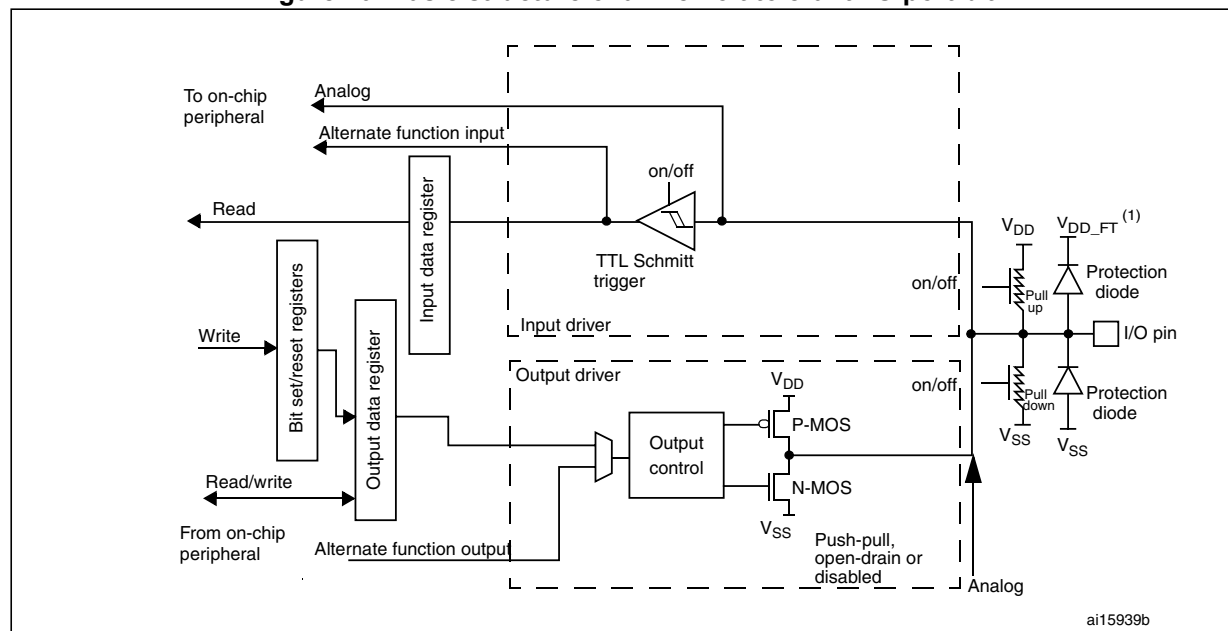


Figure 40. Basic structure of a five-volt tolerant I/O port bit



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

Table 67. Port bit configuration table⁽¹⁾

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [1:0]		PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
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.

11.3.1 General-purpose I/O (GPIO)

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

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

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.

11.3.2 I/O pin alternate function multiplexer and mapping

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

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

- After reset 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, you have to proceed as follows:

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

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

11.3.3 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.

11.3.4 I/O port data registers

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

See [Section 11.4.5: GPIO port input data register \(GPIOx_IDR\) \(x = A..H\)](#) and [Section 11.4.6: GPIO port output data register \(GPIOx_ODR\) \(x = A..H\)](#) for the register descriptions.

11.3.5 I/O data bitwise handling

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

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

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

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

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

11.3.6 GPIO locking mechanism

by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH).

The LOCK sequence (refer to [Section 11.4.8: GPIO port configuration lock register \(GPIOx_LCKR\)](#)) can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details please refer to LCKR register description in [Section 11.4.8: GPIO port configuration lock register \(GPIOx_LCKR\)](#).

11.3.7 I/O alternate function input/output

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

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

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

11.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode. [Section 14.2: Extended interrupts and events controller \(EXTI\)](#) and to [Section 14.2.3: Wakeup event management](#).

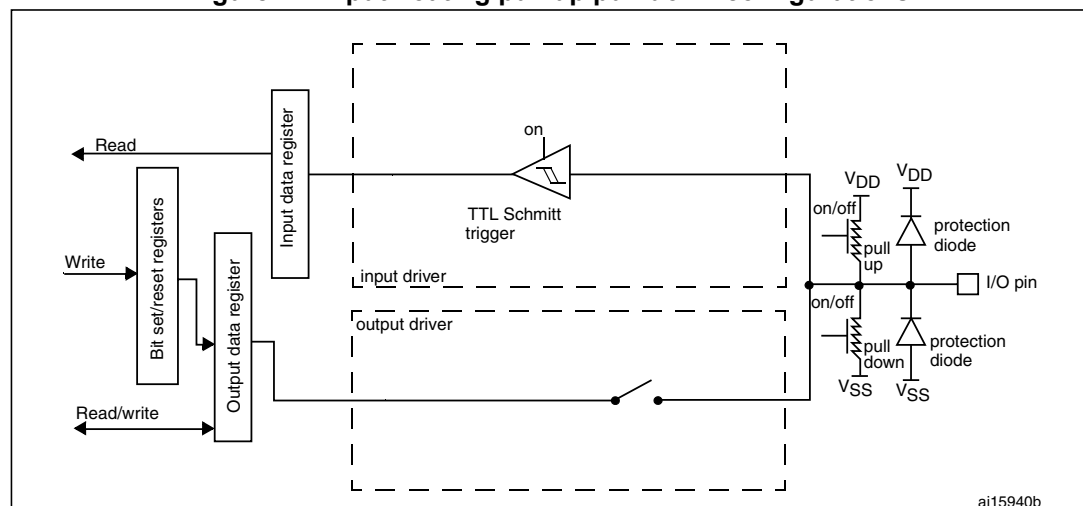
11.3.9 Input configuration

When the I/O port is programmed as input:

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

[Figure 41](#) shows the input configuration of the I/O port bit.

Figure 41. Input floating/pull up/pull down configurations



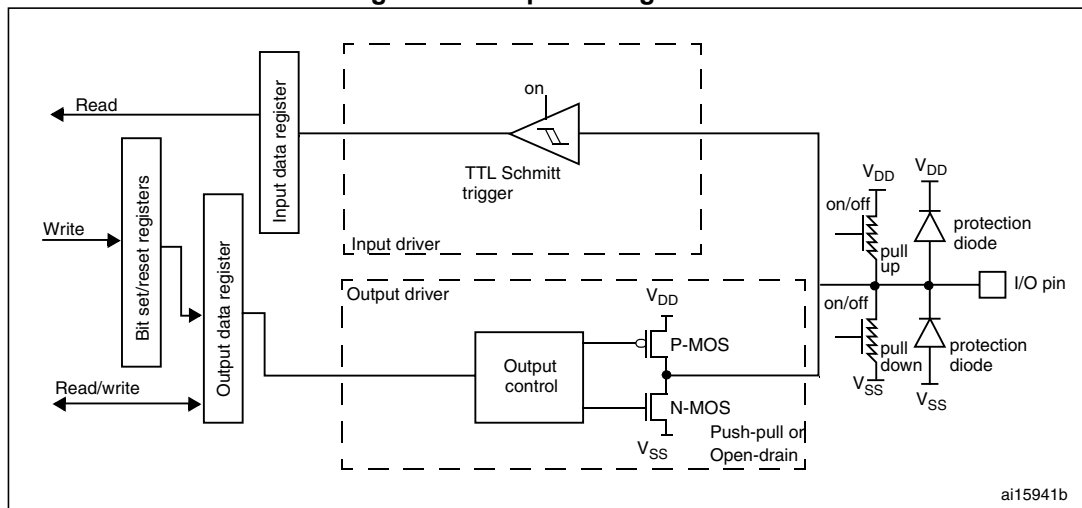
11.3.10 Output configuration

When the I/O port is programmed as output:

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

Figure 42 shows the output configuration of the I/O port bit.

Figure 42. Output configuration



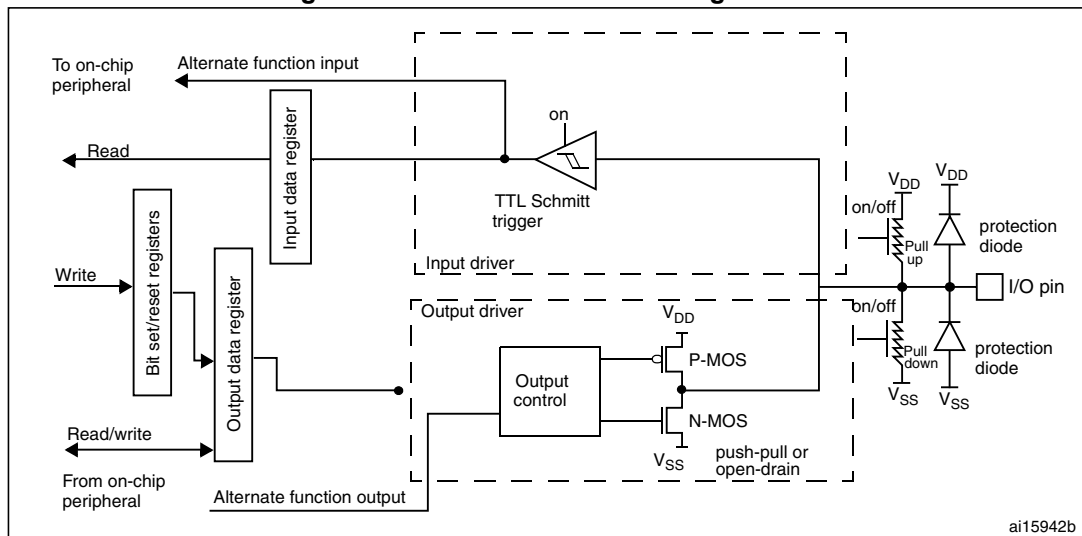
11.3.11 Alternate function configuration

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

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

Figure 43 shows the Alternate function configuration of the I/O port bit.

Figure 43. Alternate function configuration



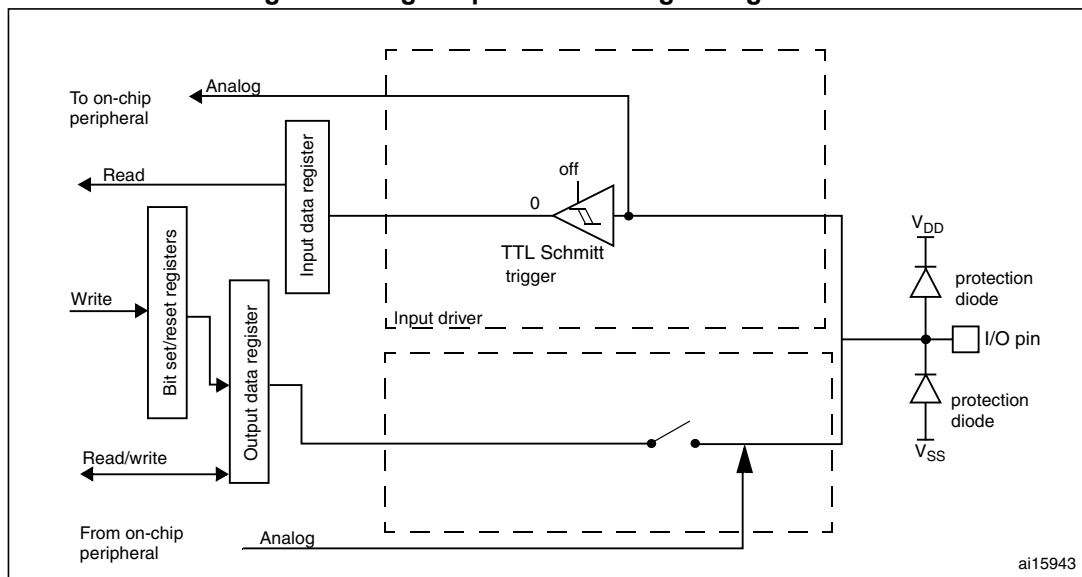
11.3.12 Analog configuration

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

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

Figure 44 shows the high-impedance, analog-input configuration of the I/O port bit.

Figure 44. High impedance-analog configuration



11.3.13 Using the HSE or LSE oscillator pins as GPIOs

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

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

When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

11.3.14 Using the GPIO pins in the RTC supply domain

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

For details about I/O control by the RTC, refer to [Section 27.3: RTC functional description on page 724](#).

11.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to [Table 68](#).

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

11.4.1 GPIO port mode register (GPIOx_MODER) (x = A..H)

Address offset: 0x00

Reset values:

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
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
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

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

00: Input mode (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode

11.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..H)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

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

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

0: Output push-pull (reset state)

1: Output open-drain

11.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..H)

Address offset: 0x08

Reset value:

- 0xC000 0000 for port A
- 0x0000 00C0 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	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
OSPEEDR7 [1:0]		OSPEEDR6 [1:0]		OSPEEDR5 [1:0]		OSPEEDR4 [1:0]		OSPEEDR3 [1:0]		OSPEEDR2 [1:0]		OSPEEDR1 [1:0]		OSPEEDR0 [1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

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

x0: Low speed

01: Medium speed

11: High speed

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

11.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..H)

Address offset: 0x0C

Reset values:

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
r/w	r/w	r/w	r/w	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
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[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 2y+1:2y **PUPDRy[1:0]**: Port x configuration bits (y = 0..15)

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

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

11.4.5 GPIO port input data register (GPIOx_IDR) (x = A..H)

Address offset: 0x10

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

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

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

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

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

11.4.6 GPIO port output data register (GPIOx_ODR) (x = A..H)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bits 15:0 **ODRy**: Port output data bit (y = 0..15)

These bits can be read and written by software.

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

11.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..H)

Address offset: 0x18

Reset value: 0x0000 0000

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

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

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

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

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

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

11.4.8 GPIO port configuration lock register (GPIOx_LCKR)

x= A, B and D in STM32F302xB/C devices, x= A, B, C, D and F in STM32F302x6/8 devices and x = A, B, C, D, E, F, G, H in STM32F302xD/E devices.

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next reset.

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

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

Address offset: 0x1C

Reset value: 0x0000 0000

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

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

Bit 16 **LCKK**: Lock key

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

0: Port configuration lock key not active

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

LOCK key write sequence:

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

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

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

RD LCKR

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

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

Any error in the lock sequence aborts the lock.

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

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

These bits are read/write but can only be written when the LCKK bit is '0'.

0: Port configuration not locked

1: Port configuration locked

11.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A..H)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]			
r/w	r/w	r/w	r/w	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
AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **AFRy[3:0]**: Alternate function selection for port x pin y (y = 0..7)

These bits are written by software to configure alternate function I/Os

11.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A..H)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]			
r/w	r/w	r/w	r/w	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
AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRy[3:0]**: Alternate function selection for port x pin y (y = 8..15)

These bits are written by software to configure alternate function I/Os

11.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A..H)

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

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

These bits are write-only. A read to these bits returns the value 0x0000

0: No action on the corresponding ODRx bit

1: Reset the corresponding ODRx bit

11.4.12 GPIO register map

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

Table 68. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	1 0		1 0		1 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0	
0x00	GPIOB_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		1 0		1 0		0 0		0 0		0 0	
0x00	GPIOx_MODER (where x = C..F)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0	
0x04	GPIOx_OTYPER (where x = A..F)	Res.	Res.	Res.	Res.	Res.	Res.	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	GPIOA_OSPEEDR	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0 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 0	
0x08	GPIOB_OSPEEDR	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		1 1		0 0		0 0		0 0	
0x08	GPIOx_OSPEEDR (where x = C..F)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0	
0x0C	GPIOA_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0 1		1 0		0 1		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0		0 0	

Table 68. GPIO register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x0C	GPIOB_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
0x10	GPIOx_IDR (where x = A..F)	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0	
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x14	GPIOx_ODR (where x = A..F)	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x18	GPIOx_BSRR (where x = A..F)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x1C	GPIOx_LCKR (where x = see ⁽¹⁾)	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0	
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20	GPIOx_AFR1 (where x = A..F)	AFRLAFR7[3:0]			AFRLAFR6[3:0]			AFRLAFR5[3:0]			AFRLAFR4[3:0]			AFRLAFR3[3:0]			AFRLAFR2[3:0]			AFRLAFR1[3:0]			AFRLAFR0[3:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOx_AFRH (where x = A..F)	AFRHAFR15[3:0]			AFRHAFR14[3:0]			AFRHAFR13[3:0]			AFRHAFR12[3:0]			AFRHAFR11[3:0]			AFRHAFR10[3:0]			AFRHAFR9[3:0]			AFRHAFR8[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

1. A, B and D in STM32F302xB/C, A, B, C, D and F in STM32F302x6/8.

Refer to [Section 3.2.2 on page 48](#) for the register boundary addresses.

12 System configuration controller (SYSCFG)

The STM32F302xx devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Enabling/disabling I²C Fm+ on some I/O ports
- Remapping some DMA trigger sources from TIM16, TIM17, TIM6, DAC1_CH1 and ADC2 to different DMA channels
- Remapping the memory located at the beginning of the code area
- Managing the external interrupt line connection to the GPIOs
- Remapping TIM1 ITR3 source
- Remapping DAC1 triggers
- Managing robustness feature
- Configuring encoder mode

12.1 SYSCFG registers

12.1.1 SYSCFG configuration register 1 (SYSCFG_CFGR1)

This register is used for specific configurations on memory remap.

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

After reset these bits take the value selected by the BOOT pin (BOOT0) and by the option bit (BOOT1).

Address offset: 0x00

Reset value: 0x7C00 000X (X is the memory mode selected by the BOOT0 pin and BOOT1 option bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPU_IE[5..0]						Res	I2C3_FMP	ENCODER_MODE	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP	
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	TIM6_DAC1_DMA_RMP	TIM17_DMA_RMP	TIM16_DMA_RMP	Res	Res	ADC2_DMA_RMP ⁽¹⁾	DAC_TRIG_RMP ⁽¹⁾	TIM1_ITR3_RMP	USB_IT_RMP	Res	Res	MEM_MODE ⁽²⁾	MEM_MODE	
		rw	rw	rw			rw	rw	rw	rw				rw	rw

1. These bits are reserved in STM32F302x6/x8

2. Only for STM32F302xD/E devices

- Bits 31:26 **FPU_IE[5..0]**: Floating Point Unit interrupts enable bits
 FPU_IE[5]: Inexact interrupt enable
 FPU_IE[4]: Input normal interrupt enable
 FPU_IE[3]: Overflow interrupt enable
 FPU_IE[2]: underflow interrupt enable
 FPU_IE[1]: Divide-by-zero interrupt enable
 FPU_IE[0]: Invalid operation interrupt enable
- Bit 25 Reserved, must be kept at reset value.
- Bit 24 **I2C3_FMP**: I2C3 fast mode Plus driving capability activation (STM32F302x6/8 devices only)
 This bit is set and cleared by software. It enables the Fm+ on I2C3 pins selected through AF selection bits.
 0: Fm+ mode is not enabled on I2C3 pins selected through AF selection bits
 1: Fm+ mode is enabled on I2C3 pins selected through AF selection bits.
- Bits 23:22 **ENCODER_MODE**: Encoder mode
 This bit is set and cleared by software.
 00: No redirection.
 01: TIM2 IC1 and TIM2 IC2 are connected to TIM15 IC1 and TIM15 IC2 respectively.
 10: TIM3 IC1 and TIM3 IC2 are connected to TIM15 IC1 and TIM15 IC2 respectively (STM32F302xB/C devices only).
 11: TIM4 IC1 and TIM4 IC2 are connected to TIM15 IC1 and TIM15 IC2 respectively (STM32F302xB/C devices only).
- Bit 21 **I2C2_FMP**: I2C2 fast mode Plus driving capability activation
 This bit is set and cleared by software. It enables the Fm+ on I2C2 pins selected through AF selection bits.
 0: Fm+ mode is not enabled on I2C2 pins selected through AF selection bits
 1: Fm+ mode is enabled on I2C2 pins selected through AF selection bits.
- Bit 20 **I2C1_FMP**: I2C1 Fm+ driving capability activation
 This bit is set and cleared by software. It enables the Fm+ on I2C1 pins selected through AF selection bits.
 0: Fm+ mode is not enabled on I2C1 pins selected through AF selection bits
 1: Fm+ mode is enabled on I2C1 pins selected through AF selection bits.
- Bits 19:16 **I2C_PbX_FMP**: Fm+ driving capability activation on the pad
 These bits are set and cleared by software. Each bit enables I²C Fm+ mode for PB6, PB7, PB8, and PB9 I/Os.
 0: PBx pin operates in standard mode (Sm), x = 6..9
 1: I²C Fm+ mode enabled on PBx pin, and the Speed control is bypassed.
- Bit 13 **TIM6_DAC1_CH1_DMA_RMP**: TIM6 and DAC channel1 DMA remap
 This bit is set and cleared by software. It controls the remapping of TIM6 (UP) and DAC channel1 DMA request.
 0: No remap (TIM6_UP and DAC_CH1 DMA requests mapped on DMA2 channel 3 in STM32F302xB/C)
 1: Remap (TIM6_UP and DAC_CH1 DMA requests mapped on DMA1 channel 3)
Note: In STM32F302x6/8, this bit must be set as there is no DMA2 in these products.
- Bit 12 **TIM17_DMA_RMP**: TIM17 DMA request remapping bit
 This bit is set and cleared by software. It controls the remapping of TIM17 DMA request.
 0: No remap (TIM17_CH1 and TIM17_UP DMA requests mapped on DMA1 channel 1)
 1: Remap (TIM17_CH1 and TIM17_UP DMA requests mapped on DMA1 channel 7)

Bit 11 **TIM16_DMA_RMP**: TIM16 DMA request remapping bit

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

- 0: No remap (TIM16_CH1 and TIM16_UP DMA requests mapped on DMA1 channel 3)
- 1: Remap (TIM16_CH1 and TIM16_UP DMA requests mapped on DMA1 channel 6)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **ADC2_DMA_RMP**: ADC2 DMA remapping bit (STM32F302xB/C/D/E devices only)

This bit is set and cleared by software. It controls the remapping of ADC24 DMA requests.

- 0: No remap (ADC24 DMA requests mapped on DMA2 channels 1 and 2)
- 1: Remap (ADC24 DMA requests mapped on DMA2 channels 3 and 4)

Bit 7 **DAC1_TRIG_RMP**: DAC trigger remap (when TSEL = 001) (STM32F302xB/C/D/E devices only)

This bit is set and cleared by software. It controls the mapping of the DAC trigger source.

- 0: No remap
- 1: Remap (DAC trigger is TIM3_TRGO)

Bit 6 **TIM1_ITR3_RMP**: Timer 1 ITR3 selection

This bit is set and cleared by software. It controls the mapping of TIM1 ITR3.

- 0: No remap (TIM1_ITR3 = TIM4_TRGO in STM32F302xB/C/D/E devices)
- 1: Remap (TIM1_ITR3 = TIM17_OC)

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **MEM_MODE**: Memory mapping selection bits

This bit is set and cleared by software. It controls the memory internal mapping at address 0x0000 0000. After reset these bits take on the memory mapping selected by BOOT0 pin and BOOT1 option bit.

- 0x0: Main Flash memory mapped at 0x0000 0000
- 001: System Flash memory mapped at 0x0000 0000
- 011: Embedded SRAM (on the D-Code bus) mapped at 0x0000 0000
- 1xx: FMC Bank (Only the first two banks) (Available on STM32F302xD/E only).

12.1.2 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

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
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **EXTI3[3:0]**: EXTI 3 configuration bits

These bits are written by software to select the source input for the EXTI3 external interrupt.

x000: PA[3] pin
 x001: PB[3] pin
 x010: PC[3] pin
 x011: PD[3] pin
 x100: PE[3] pin
 other configurations: reserved

Bits 11:8 **EXTI2[3:0]**: EXTI 2 configuration bits

These bits are written by software to select the source input for the EXTI2 external interrupt.

x000: PA[2] pin
 x001: PB[2] pin
 x010: PC[2] pin
 x011: PD[2] pin
 x100: PE[2] pin
 x101: PF[2] pin
 other configurations: reserved

Bits 7:4 **EXTI1[3:0]**: EXTI 1 configuration bits

These bits are written by software to select the source input for the EXTI1 external interrupt.

x000: PA[1] pin
 x001: PB[1] pin
 x010: PC[1] pin
 x011: PD[1] pin
 x100: PE[1] pin
 x101: PF[1] pin
 other configurations: reserved

Bits 3:0 **EXTI0[3:0]**: EXTI 0 configuration bits

These bits are written by software to select the source input for the EXTI0 external interrupt.

x000: PA[0] pin
 x001: PB[0] pin
 x010: PC[0] pin
 x011: PD[0] pin
 x100: PE[0] pin
 x101: PF[0] pin
 other configurations: reserved

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

12.1.3 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)

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
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **EXTI7[3:0]**: EXTI 7 configuration bits

These bits are written by software to select the source input for the EXTI7 external interrupt.

x000: PA[7] pin

x001: PB[7] pin

x010: PC[7] pin

x011: PD[7] pin

x100: PE[7] pin

Other configurations: reserved

Bits 11:8 **EXTI6[3:0]**: EXTI 6 configuration bits

These bits are written by software to select the source input for the EXTI6 external interrupt.

x000: PA[6] pin

x001: PB[6] pin

x010: PC[6] pin

x011: PD[6] pin

x100: PE[6] pin

x101: PF[6] pin

Other configurations: reserved

Bits 7:4 **EXTI5[3:0]**: EXTI 5 configuration bits

These bits are written by software to select the source input for the EXTI5 external interrupt.

x000: PA[5] pin

x001: PB[5] pin

x010: PC[5] pin

x011: PD[5] pin

x100: PE[5] pin

x101: PF[5] pin

Other configurations: reserved

Bits 3:0 **EXTI4[3:0]**: EXTI 4 configuration bits

These bits are written by software to select the source input for the EXTI4 external interrupt.

x000: PA[4] pin

x001: PB[4] pin

x010: PC[4] pin

x011: PD[4] pin

x100: PE[4] pin

x101: PF[4] pin

Other configurations: reserved

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

12.1.4 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
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 **EXTI11[3:0]**: EXTI 11 configuration bits

These bits are written by software to select the source input for the EXTI11 external interrupt.

- x000: PA[11] pin
- x001: PB[11] pin
- x010: PC[11] pin
- x011: PD[11] pin
- x100: PE[11] pin
- other configurations: reserved



Bits 11:8 **EXTI10[3:0]**: EXTI 10 configuration bits

These bits are written by software to select the source input for the EXTI10 external interrupt.

x000: PA[10] pin
x001: PB[10] pin
x010: PC[10] pin
x011: PD[10] pin
x100: PE[10] pin
x101: PF[10] pin
other configurations: reserved

Bits 7:4 **EXTI9[3:0]**: EXTI 9 configuration bits

These bits are written by software to select the source input for the EXTI9 external interrupt.

x000: PA[9] pin
x001: PB[9] pin
x010: PC[9] pin
x011: PD[9] pin
x100: PE[9] pin
x101: PF[9] pin
other configurations: reserved

Bits 3:0 **EXTI8[3:0]**: EXTI 8 configuration bits

These bits are written by software to select the source input for the EXTI8 external interrupt.

x000: PA[8] pin
x001: PB[8] pin
x010: PC[8] pin
x011: PD[8] pin
x100: PE[8] pin
other configurations: reserved

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

12.1.5 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)

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
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **EXTI15[3:0]**: EXTI15 configuration bits

These bits are written by software to select the source input for the EXTI15 external interrupt.

x000: PA[15] pin
 x001: PB[15] pin
 x010: PC[15] pin
 x011: PD[15] pin
 x100: PE[15] pin
 Other configurations: reserved

Bits 11:8 **EXTI14[3:0]**: EXTI14 configuration bits

These bits are written by software to select the source input for the EXTI14 external interrupt.

x000: PA[14] pin
 x001: PB[14] pin
 x010: PC[14] pin
 x011: PD[14] pin
 x100: PE[14] pin
 Other configurations: reserved

Bits 7:4 **EXTI13[3:0]**: EXTI13 configuration bits

These bits are written by software to select the source input for the EXTI13 external interrupt.

x000: PA[13] pin
 x001: PB[13] pin
 x010: PC[13] pin
 x011: PD[13] pin
 x100: PE[13] pin
 Other configurations: reserved

Bits 3:0 **EXTI12[3:0]**: EXTI12 configuration bits

These bits are written by software to select the source input for the EXTI12 external interrupt.

x000: PA[12] pin
 x001: PB[12] pin
 x010: PC[12] pin
 x011: PD[12] pin
 x100: PE[12] pin
 Other configurations: reserved

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

12.1.6 SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset: 0x18

System reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	SRAM_PEF ⁽¹⁾	Res	Res	Res	BYP_ADDR_PAR ⁽¹⁾	Res	PVD_LOCK	SRAM_PARITY_LOCK ⁽¹⁾	LOCKUP_LOCK
							rc_w1				rw		rw	rw	rw

1. Available in STM32F302xB/xC/D/E only.

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

Bit 8 **SRAM_PEF**: SRAM parity error flag (STM32F302xB/C/D/E devices only)

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 7:5 Reserved, must be kept at reset value

Bit 4 **BYP_ADDR_PAR**: Bypass address bit 29 in parity calculation (STM32F302xB/C/D/E devices only)

This bit is set by software and cleared by a system reset. It is used to prevent an unwanted parity error when the user writes a code in the RAM at address 0x2XXXXXXX (address in the address range 0x20000000-0x20002000) and then executes the code from RAM at boot (RAM is remapped at address 0x00). In this case, a read operation will be performed from the range 0x00000000-0x00002000 resulting in a parity error (the parity on the address is different).

0: The ramload operation is performed taking into consideration bit 29 of the address when the parity is calculated.

1: The ramload operation is performed without taking into consideration bit 29 of the address when the parity is calculated.

Bit 3 Reserved, must be kept at reset value

Bit 2 PVD_LOCK: PVD lock enable bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the PVD connection to 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 (STM32F302xB/C/D/E devices only)

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 inputs.

0: SRAM parity error signal disconnected from TIM1/15/16/17 Break inputs

1: SRAM parity error signal connected to TIM1/15/16/17 Break inputs

Bit 0 LOCKUP_LOCK: Cortex[®]-M4 LOCKUP (Hardfault) output enable bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the connection of Cortex[®]-M4 LOCKUP (Hardfault) output to TIM1/15/16/17 Break input.

0: Cortex[®]-M4 LOCKUP output disconnected from TIM1/15/16/17 Break inputs.

1: Cortex[®]-M4 LOCKUP output connected to TIM1/15/16/17 Break inputs

The following table gives the SYSCFG register map and the 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	SYSCFG_CFGR1	FPU_IE[5..0]								Res		I2C3_FMP		ENCODER_MODE [1:0]		I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP	Res		Res		ADC24_DMA_RMP	DAC_TRIG_RMP	TIM1_ITR3_RMP	USB_IT_RMP	Res		Res	Res	MEM_MODE	
	Reset value	1	1	1	1	1	0				0	0	0	0	0	0	0				0	0	0			0	0	0	0				X	X	
0x08	SYSCFG_EXTICR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI3[3:0]			EXTI2[3:0]			EXTI1[3:0]			EXTI0[3:0]									
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0C	SYSCFG_EXTICR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI7[3:0]			EXTI6[3:0]			EXTI5[3:0]			EXTI4[3:0]									
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x10	SYSCFG_EXTICR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI11[3:0]			EXTI10[3:0]			EXTI9[3:0]			EXTI8[3:0]									
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x14	SYSCFG_EXTICR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI15[3:0]			EXTI14[3:0]			EXTI13[3:0]			EXTI12[3:0]									
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x18	SYSCFG_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM_PEF		Res.	Res.	Res.	BYP_ADDR_PAR		Res.	PVD_LOCK	SRAM_PARITY_LOCK	LOCKUP_LOCK
	Reset value																0	0	0	0	0	0	0	0	0			0	0	0	0	0	0		

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

13 Direct memory access controller (DMA)

13.1 Introduction

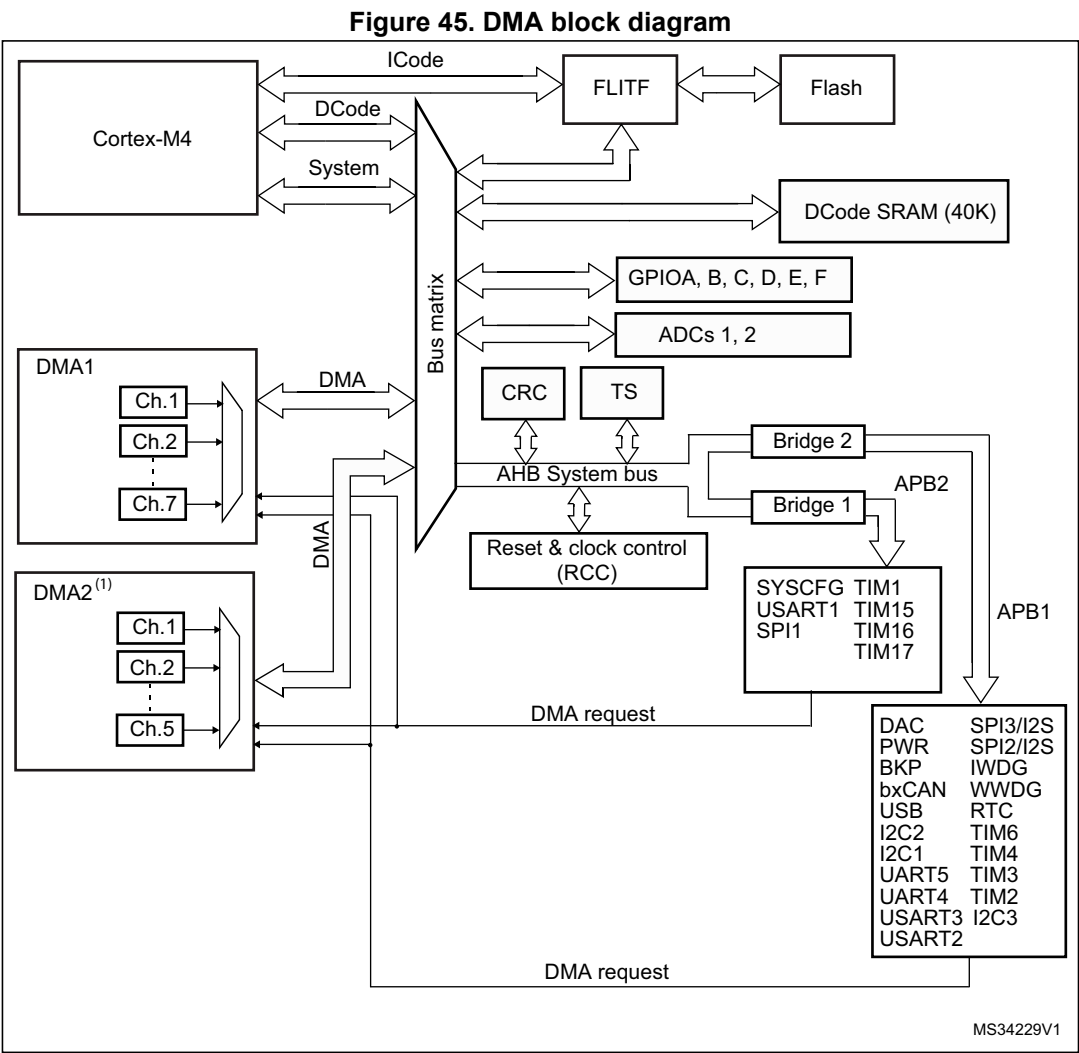
Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory as well as memory to memory. Data can be quickly moved by DMA without any CPU actions. This keeps CPU resources free for other operations.

The STM32F302xB/C/D/E devices have two DMA controllers with 12 channels in total, The STM32F302x6/8 has 1 DMA controller with 7 channels. Each channel is dedicated to managing memory access requests from one or more peripherals. Each has an arbiter for handling the priority between DMA requests.

13.2 DMA main features

- 12 independently configurable channels (requests) on STM32F302xB/C/D/E and STM32F302x6/8 devices and 7 independently configurable channels (requests) on STM32F302x6/8 devices
- Each channel is connected to dedicated hardware DMA requests, software trigger is also supported on each channel. This configuration is done by software.
- Priorities between requests from the DMA channels are software programmable (4 levels consisting of *very high*, *high*, *medium*, *low*) or hardware in case of equality (request 1 has priority over request 2, etc.)
- Independent source and destination transfer size (byte, half word, word), emulating packing and unpacking. Source/destination addresses must be aligned on the data size.
- Support for circular buffer management
- 3 event flags (DMA Half Transfer, DMA Transfer complete and DMA Transfer Error) logically ORed together in a single interrupt request for each channel
- Memory-to-memory transfer
- Peripheral-to-memory and memory-to-peripheral, and peripheral-to-peripheral transfers
- Access to Flash, SRAM, APB and AHB peripherals as source and destination
- Programmable number of data to be transferred: up to 65535

The block diagram is shown in the following figure.



1. DMA2, SPI1, TIM3, TIM4, UART4, UART5 and ADC2 are not available in STM32F302x6/8 devices.
2. I2C3 is not available in STM32F302xB/C devices.

13.3 DMA implementation

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

Table 70. DMA implementation

Feature	DMA1	DMA2 ⁽¹⁾
Number of DMA channels	7	5

1. DMA2 is not available on STM32F302x6/8 devices.

13.4 DMA functional description

The DMA controller performs direct memory transfer by sharing the system bus with the Cortex-M4[®]F core. The DMA request may stop the CPU access to the system bus for some bus cycles, when the CPU and DMA are targeting the same destination (memory or peripheral). The bus matrix implements round-robin scheduling, thus ensuring at least half of the system bus bandwidth (both to memory and peripheral) for the CPU.

13.4.1 DMA transactions

After an event, the peripheral sends a request signal to the DMA Controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA Controller accesses the peripheral, an Acknowledge is sent to the peripheral by the DMA Controller. The peripheral releases its request as soon as it gets the Acknowledge from the DMA Controller. Once the request is de-asserted by the peripheral, the DMA Controller release the Acknowledge. If there are more requests, the peripheral can initiate the next transaction.

In summary, each DMA transfer consists of three operations:

- The loading of data from the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register
- The storage of the data loaded to the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register
- The post-decrementing of the DMA_CNDTRx register, which contains the number of transactions that have still to be performed.

13.4.2 Arbiter

The arbiter manages the channel requests based on their priority and launches the peripheral/memory access sequences.

The priorities are managed in two stages:

- Software: each channel priority can be configured in the DMA_CCRx register. There are four levels:
 - Very high priority
 - High priority
 - Medium priority
 - Low priority
- Hardware: if 2 requests have the same software priority level, the channel with the lowest number will get priority versus the channel with the highest number. For example, channel 2 gets priority over channel 4.

13.4.3 DMA channels

Each channel can handle DMA transfer between a peripheral register located at a fixed address and a memory address. The amount of data to be transferred (up to 65535) is programmable. The register which contains the amount of data items to be transferred is decremented after each transaction.

Programmable data sizes

Transfer data sizes of the peripheral and memory are fully programmable through the PSIZE and MSIZE bits in the DMA_CCRx register.

Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented after each transaction depending on the PINC and MINC bits in the DMA_CCRx register. If incremented mode is enabled, the address of the next transfer will be the address of the previous one incremented by 1, 2 or 4 depending on the chosen data size. The first transfer address is the one programmed in the DMA_CPARx/DMA_CMARx registers. During transfer operations, these registers keep the initially programmed value. The current transfer addresses (in the current internal peripheral/memory address register) are not accessible by software.

If the channel is configured in non-circular mode, no DMA request is served after the last transfer (that is once the number of data items to be transferred has reached zero). In order to reload a new number of data items to be transferred into the DMA_CNDTRx register, the DMA channel must be disabled.

Note: If a DMA channel is disabled, the DMA registers are not reset. The DMA channel registers (DMA_CCRx, DMA_CPARx and DMA_CMARx) retain the initial values programmed during the channel configuration phase.

In circular mode, after the last transfer, the DMA_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address values from the DMA_CPARx/DMA_CMARx registers.

Channel configuration procedure

The following sequence should be followed to configure a DMA channel x (where x is the channel number).

1. Set the peripheral register address in the DMA_CPARx register. The data will be moved from/ to this address to/ from the memory after the peripheral event.
2. Set the memory address in the DMA_CMARx register. The data will be written to or read from this memory after the peripheral event.
3. Configure the total number of data to be transferred in the DMA_CNDTRx register. After each peripheral event, this value will be decremented.
4. Configure the channel priority using the PL[1:0] bits in the DMA_CCRx register
5. Configure data transfer direction, circular mode, peripheral & memory incremented mode, peripheral & memory data size, and interrupt after half and/or full transfer in the DMA_CCRx register
6. Activate the channel by setting the ENABLE bit in the DMA_CCRx register.

As soon as the channel is enabled, it can serve any DMA request from the peripheral connected on the channel.

Once half of the bytes are transferred, the half-transfer flag (HTIF) is set and an interrupt is generated if the Half-Transfer Interrupt Enable bit (HTIE) is set. At the end of the transfer, the Transfer Complete Flag (TCIF) is set and an interrupt is generated if the Transfer Complete Interrupt Enable bit (TCIE) is set.

Circular mode

Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_CCRx register. When circular mode is activated, the number of data to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

Memory-to-memory mode

The DMA channels can also work without being triggered by a request from a peripheral. This mode is called Memory to Memory mode.

If the MEM2MEM bit in the DMA_CCRx register is set, then the channel initiates transfers as soon as it is enabled by software by setting the Enable bit (EN) in the DMA_CCRx register. The transfer stops once the DMA_CNDTRx register reaches zero. Memory to Memory mode may not be used at the same time as Circular mode.

13.4.4 Programmable data width, data alignment and endians

When PSIZE and MSIZE are not equal, the DMA performs some data alignments as described in [Table 71: Programmable data width & endian behavior \(when bits PINC = MINC = 1\)](#).

Table 71. Programmable data width & endian behavior (when bits PINC = MINC = 1)

Source port width	Destination port width	Number of data items to transfer (NDT)	Source content: address / data	Transfer operations	Destination content: address / data
8	8	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B1[7:0] @0x1 then WRITE B1[7:0] @0x1 3: READ B2[7:0] @0x2 then WRITE B2[7:0] @0x2 4: READ B3[7:0] @0x3 then WRITE B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8	16	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE 00B0[15:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 00B1[15:0] @0x2 3: READ B3[7:0] @0x2 then WRITE 00B2[15:0] @0x4 4: READ B4[7:0] @0x3 then WRITE 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8	32	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE 000000B0[31:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 000000B1[31:0] @0x4 3: READ B3[7:0] @0x2 then WRITE 000000B2[31:0] @0x8 4: READ B4[7:0] @0x3 then WRITE 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B2[7:0] @0x1 3: READ B5B4[15:0] @0x4 then WRITE B4[7:0] @0x2 4: READ B7B6[15:0] @0x6 then WRITE B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16	16	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE B1B0[15:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B3B2[15:0] @0x2 3: READ B5B4[15:0] @0x4 then WRITE B5B4[15:0] @0x4 4: READ B7B6[15:0] @0x6 then WRITE B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16	32	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE 0000B1B0[31:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE 0000B3B2[31:0] @0x4 3: READ B5B4[15:0] @0x4 then WRITE 0000B5B4[31:0] @0x8 4: READ B7B6[15:0] @0x6 then WRITE 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6

Table 71. Programmable data width & endian behavior (when bits PINC = MINC = 1) (continued)

Source port width	Destination port width	Number of data items to transfer (NDT)	Source content: address / data	Transfer operations	Destination content: address / data
32	8	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B4[7:0] @0x1 3: READ BBBAB9B8[31:0] @0x8 then WRITE B8[7:0] @0x2 4: READ BFBEBDBC[31:0] @0xC then WRITE BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC
32	16	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B1B0[15:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B5B4[15:0] @0x2 3: READ BBBAB9B8[31:0] @0x8 then WRITE B9B8[15:0] @0x4 4: READ BFBEBDBC[31:0] @0xC then WRITE BDBC[15:0] @0x6	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC
32	32	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B3B2B1B0[31:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B7B6B5B4[31:0] @0x4 3: READ BBBAB9B8[31:0] @0x8 then WRITE BBBAB9B8[31:0] @0x8 4: READ BFBEBDBC[31:0] @0xC then WRITE BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC

Addressing an AHB peripheral that does not support byte or halfword write operations

When the DMA initiates an AHB byte or halfword write operation, the data are duplicated on the unused lanes of the HWDATA[31:0] bus. So when the used AHB slave peripheral does not support byte or halfword write operations (when HSIZE is not used by the peripheral) *and* does not generate any error, the DMA writes the 32 HWDATA bits as shown in the two examples below:

- To write the halfword “0xABCD”, the DMA sets the HWDATA bus to “0xABCDABCD” with HSIZE = HalfWord
- To write the byte “0xAB”, the DMA sets the HWDATA bus to “0xABABABAB” with HSIZE = Byte

Assuming that the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take the HSIZE data into account, it will transform any AHB byte or halfword operation into a 32-bit APB operation in the following manner:

- an AHB byte write operation of the data “0xB0” to 0x0 (or to 0x1, 0x2 or 0x3) will be converted to an APB word write operation of the data “0xB0B0B0B0” to 0x0
- an AHB halfword write operation of the data “0xB1B0” to 0x0 (or to 0x2) will be converted to an APB word write operation of the data “0xB1B0B1B0” to 0x0

For instance, if you want to write the APB backup registers (16-bit registers aligned to a 32-bit address boundary), you must configure the memory source size (MSIZE) to “16-bit” and the peripheral destination size (PSIZE) to “32-bit”.

13.4.5 Error management

A DMA transfer error can be generated by reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or a write access, the faulty channel is automatically disabled through a hardware clear of its EN bit in the corresponding Channel configuration register (DMA_CCRx). The channel's transfer error interrupt flag (TEIF) in the DMA_IFR register is set and an interrupt is generated if the transfer error interrupt enable bit (TEIE) in the DMA_CCRx register is set.

13.4.6 DMA interrupts

An interrupt can be produced on a Half-transfer, Transfer complete or Transfer error for each DMA channel. Separate interrupt enable bits are available for flexibility.

Table 72. DMA interrupt requests

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

13.4.7 DMA request mapping

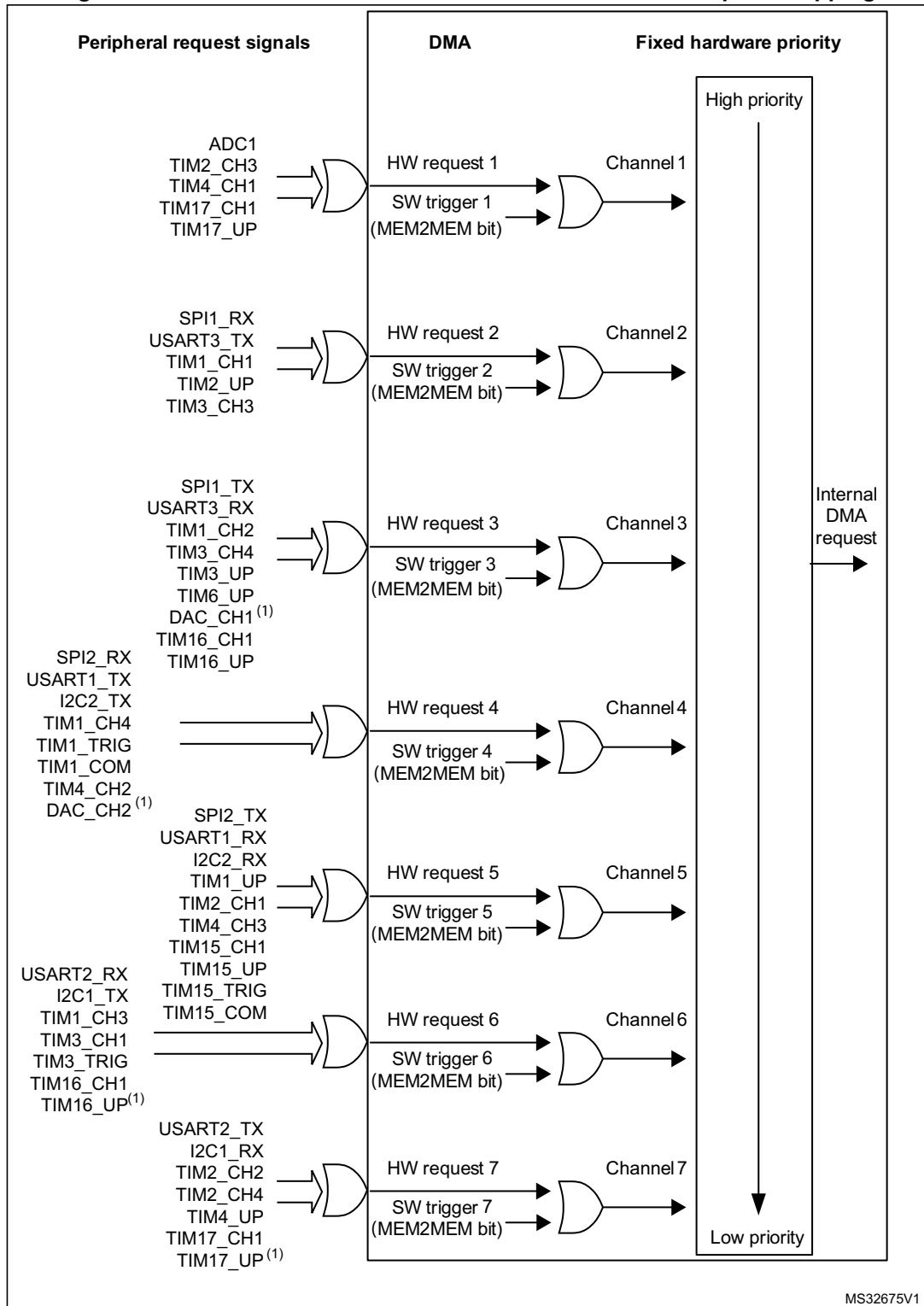
DMA1 controller

The hardware requests from the peripherals (TIMx(x=1...4, 6, 15..17), ADC1, ADC2, SPI1 (STM32F302x6/8 only), SPI2/I2S, SPI3/I2S, I2Cx(x=1,2,3), DAC1_Channel[1] and USARTx (x=1..3)) are simply logically ORed before entering the DMA1. This means that on one channel, only one request must be enabled at a time. Refer to [Figure 46: STM32F302xB/C/D/E and STM32F302x6/8 DMA1 request mapping](#) and [Figure 47: STM32F302x6/8 DMA1 request mapping](#).

Refer to Figure 46: STM32F302xB/C/D/E and STM32F302x6/8 DMA1 request mapping and Figure 47: STM32F302x6/8 DMA1 request mapping.

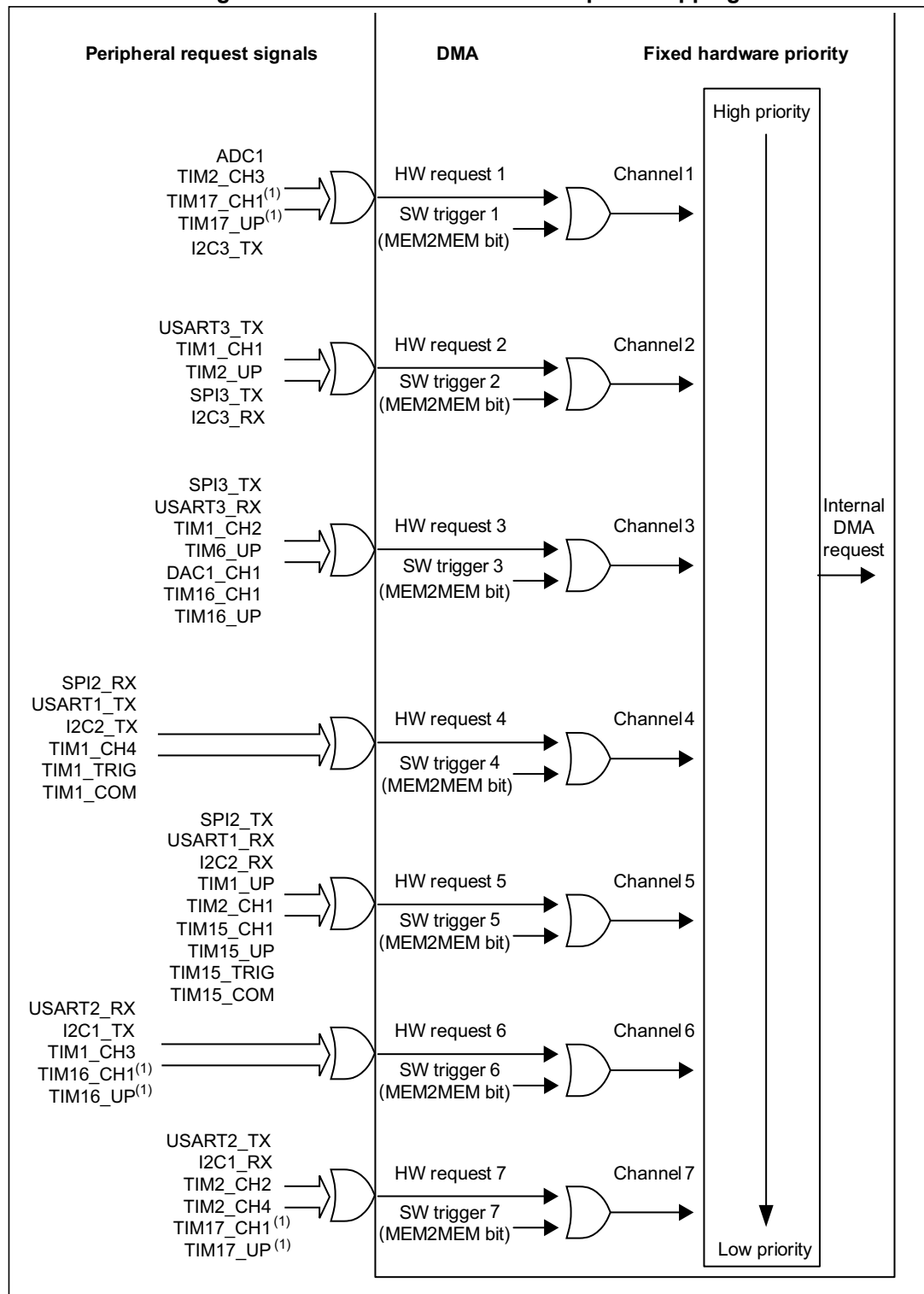
The peripheral DMA requests can be independently activated/de-activated by programming the DMA control bit in the registers of the corresponding peripheral.

Figure 46. STM32F302xB/C/D/E and STM32F302x6/8 DMA1 request mapping



1. DMA requests are mapped on this DMA channel only if the corresponding remapping bit is set in the SYSCFG_CFGR1 register. For more details, please refer to [Section 12.1.1: SYSCFG configuration register 1 \(SYSCFG_CFGR1\) on page 233](#).

Figure 47. STM32F302x6/8 DMA1 request mapping



1. TIM16_CH1, TIM16_UP, TIM17_CH1, TIM17_UP and DMA request are mapped on this DMA channel only if the corresponding remapping bit is set in the SYSCFG_CFGR1 or SYSCFG_CFGR3 register. For more details, please refer to [Section 12.1.1: SYSCFG configuration register 1 \(SYSCFG_CFGR1\) on page 233](#).

Table 73. STM32F302xB/C/D/E summary of DMA1 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel6	Channel7
ADC	ADC1						
SPI		SPI1_RX	SP1_TX	SPI2_RX	SPI2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I2C	I2C3_TX ⁽¹⁾	I2C3_RX ⁽¹⁾		I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP
TIM6 / DAC			TIM6_UP DAC_CH1 ⁽²⁾				
TIM15					TIM15_CH1 TIM15_UP TIM15_TRIG TIM15_COM		
TIM16			TIM16_CH1 TIM16_UP			TIM16_CH1 TIM16_UP ⁽²⁾	
TIM17	TIM17_CH1 TIM17_UP						TIM17_CH1 TIM17_UP ⁽²⁾

1. Available in STM32F302xD/E only.

2. DMA request mapped on this DMA channel only if the corresponding remapping bit is set in the SYSCFG_CFGR1 register. For more details, please refer to [Section 12.1.1: SYSCFG configuration register 1 \(SYSCFG_CFGR1\) on page 233](#).

Table 74. STM32F302x6/8 summary of DMA1 requests for each channel

Peripheral	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel6	Channel7
ADC	ADC1						
SPI		SPI3_RX	SPI3_TX	SPI2_RX	SPI2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I2C	I2C3_TX	I2C3_RX		I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM6/DAC			TIM6_UP DAC_CH1 ⁽¹⁾				

Table 74. STM32F302x6/8 summary of DMA1 requests for each channel (continued)

Peripheral	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel6	Channel7
TIM15					TIM15_CH1 TIM15_UP TIM15_TRIG TIM15_COM		
TIM16			TIM16_CH1 TIM16_UP			TIM16_CH1 TIM16_UP ⁽¹⁾	
TIM17	TIM17_CH1 TIM17_UP						TIM17_CH1 TIM17_UP ⁽¹⁾

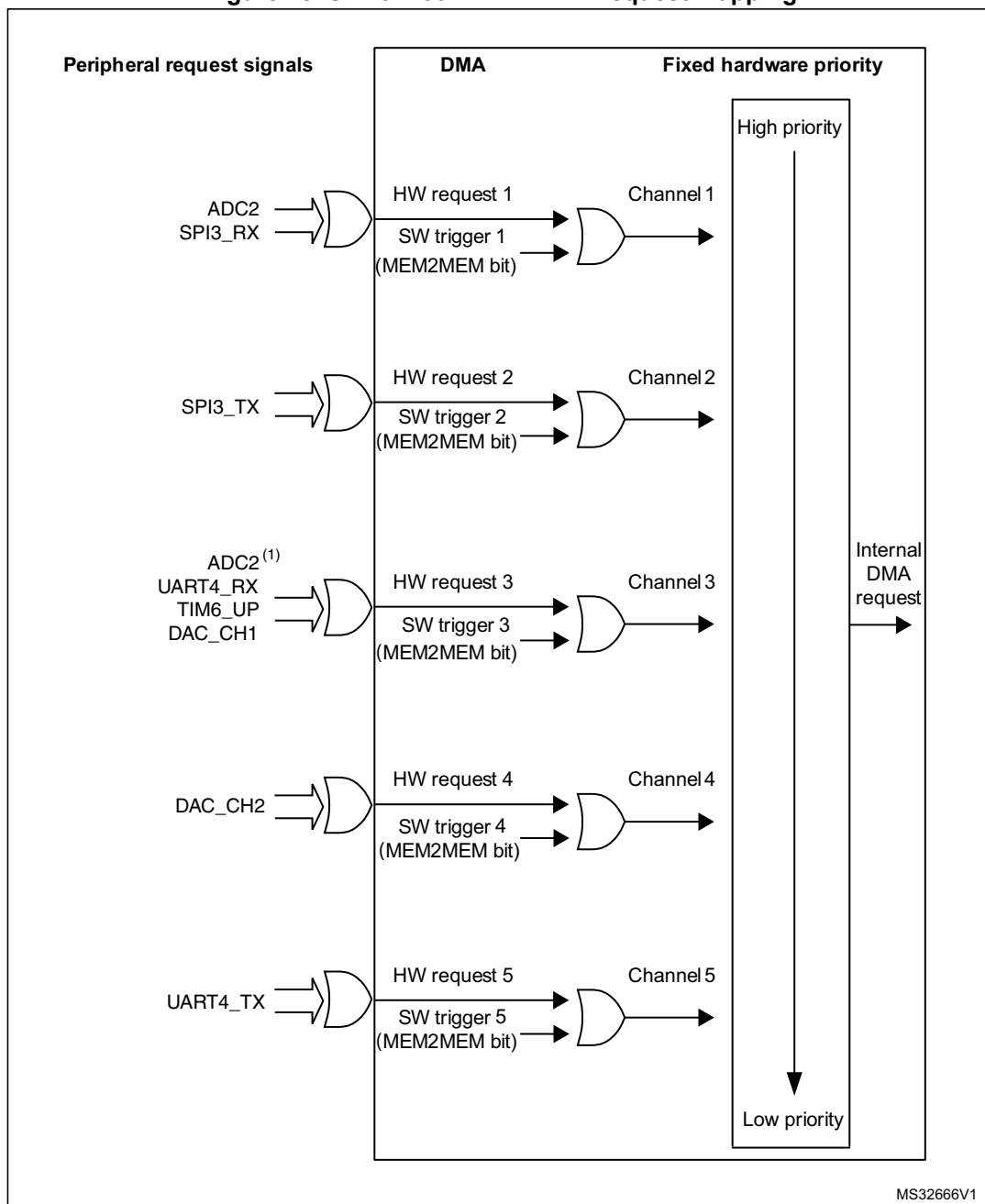
1. DMA request mapped on this DMA channel only if the corresponding remapping bit is set in the SYSCFG_CFGR1 or SYSCFGR3 register. For more details, please refer to [Section 12.1.1: SYSCFG configuration register 1 \(SYSCFG_CFGR1\) on page 233](#).

DMA2 controller (available only in STM32F302xB/C/D/E)

The five requests from the peripherals (TIMx (x= 6), ADCx (x=2), SPI/I2S3, UART4, DAC_Channel[1,2]) are simply logically ORed before entering the DMA2, this means that only one request must be enabled at a time. Refer to [Figure 48: STM32F302xB/C DMA2 request mapping](#).

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

Figure 48. STM32F302xB/C DMA2 request mapping



1. DMA request mapped on this DMA channel only if the corresponding remapping bit is set in the SYSCFG_CFGR1 register. For more details, please refer to [Section 12.1.1: SYSCFG configuration register 1 \(SYSCFG_CFGR1\) on page 233](#).

Table 75 lists the DMA requests for each channel.

Table 75. STM32F302xB/C/D/E summary of DMA2 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC	ADC2	-			-
SPI3	SPI3_RX	SPI3_TX	-	SPI4_RX ⁽¹⁾	SPI4_TX ⁽¹⁾
UART4	-	-	UART4_RX	-	UART4_TX
TIM6 / DAC	-	-	TIM6_UP DAC_CH1	-	-

1. ***Available in STM32F302xD/E only.

13.5 DMA registers

Refer to [Section 2.1 on page 42](#) for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by bytes (8-bit), half-words (16-bit) or words (32-bit).

13.5.1 DMA interrupt status register (DMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 27, 23, 19, 15, **TEIFx**: Channel x transfer error flag (x = 1..7)

11, 7, 3 This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No transfer error (TE) on channel x

1: A transfer error (TE) occurred on channel x

Bits 26, 22, 18, 14, **HTIFx**: Channel x half transfer flag (x = 1..7)

10, 6, 2 This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No half transfer (HT) event on channel x

1: A half transfer (HT) event occurred on channel x

Bits 25, 21, 17, 13, **TCIFx**: Channel x transfer complete flag (x = 1..7)

9, 5, 1 This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No transfer complete (TC) event on channel x

1: A transfer complete (TC) event occurred on channel x

Bits 24, 20, 16, 12, **GIFx**: Channel x global interrupt flag (x = 1..7)

8, 4, 0 This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No TE, HT or TC event on channel x

1: A TE, HT or TC event occurred on channel x

13.5.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTEIF 7	CHTIF 7	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
CTEIF 4	CHTIF 4	CTCIF 4	CGIF4	CTEIF 3	CHTIF 3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bits 27, 23, 19, 15, **CTEIFx**: Channel x transfer error clear (x = 1..7)

11, 7, 3 This bit is set and cleared by software.

0: No effect

1: Clears the corresponding TEIF flag in the DMA_ISR register

Bits 26, 22, 18, 14, **CHTIFx**: Channel x half transfer clear (x = 1..7)

10, 6, 2 This bit is set and cleared by software.

0: No effect

1: Clears the corresponding HTIF flag in the DMA_ISR register

Bits 25, 21, 17, 13, **CTCIFx**: Channel x transfer complete clear (x = 1..7)

9, 5, 1 This bit is set and cleared by software.

0: No effect

1: Clears the corresponding TCIF flag in the DMA_ISR register

Bits 24, 20, 16, 12, **CGIFx**: Channel x global interrupt clear (x = 1..7)

8, 4, 0 This bit is set and cleared by software.

0: No effect

1: Clears the GIF, TEIF, HTIF and TCIF flags in the DMA_ISR register

13.5.3 DMA channel x configuration register (DMA_CCRx) (x = 1..7 , where x = channel number)

Address offset: 0x08 + 0d20 × (channel number – 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2 MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **MEM2MEM**: Memory to memory mode

This bit is set and cleared by software.

0: Memory to memory mode disabled

1: Memory to memory mode enabled

Bits 13:12 **PL[1:0]**: Channel priority level

These bits are set and cleared by software.

00: Low

01: Medium

10: High

11: Very high

Bits 11:10 **MSIZE[1:0]**: Memory size

These bits are set and cleared by software.

00: 8-bits

01: 16-bits

10: 32-bits

11: Reserved

Bits 9:8 **PSIZE[1:0]**: Peripheral size

These bits are set and cleared by software.

00: 8-bits

01: 16-bits

10: 32-bits

11: Reserved

Bit 7 **MINC**: Memory increment mode

This bit is set and cleared by software.

0: Memory increment mode disabled

1: Memory increment mode enabled

Bit 6 **PINC**: Peripheral increment mode

This bit is set and cleared by software.

0: Peripheral increment mode disabled

1: Peripheral increment mode enabled

- Bit 5 **CIRC**: Circular mode
This bit is set and cleared by software.
0: Circular mode disabled
1: Circular mode enabled
- Bit 4 **DIR**: Data transfer direction
This bit is set and cleared by software.
0: Read from peripheral
1: Read from memory
- Bit 3 **TEIE**: Transfer error interrupt enable
This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled
- Bit 2 **HTIE**: Half transfer interrupt enable
This bit is set and cleared by software.
0: HT interrupt disabled
1: HT interrupt enabled
- Bit 1 **TCIE**: Transfer complete interrupt enable
This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled
- Bit 0 **EN**: Channel enable
This bit is set and cleared by software.
0: Channel disabled
1: Channel enabled

13.5.4 DMA channel x number of data register (DMA_CNDTRx) (x = 1..7, where x = channel number)

Address offset: $0x0C + 0d20 \times (\text{channel number} - 1)$

Reset value: 0x0000 0000

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

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

Bits 15:0 **NDT[15:0]**: Number of data to transfer

Number of data to be transferred (0 up to 65535). This register can only be written when the channel is disabled. Once the channel is enabled, this register is read-only, indicating the remaining bytes to be transmitted. This register decrements after each DMA transfer.

Once the transfer is completed, this register can either stay at zero or be reloaded automatically by the value previously programmed if the channel is configured in auto-reload mode.

If this register is zero, no transaction can be served whether the channel is enabled or not.

13.5.5 DMA channel x peripheral address register (DMA_CPARx) (x = 1..7, where x = channel number)

Address offset: $0x10 + 0d20 \times (\text{channel number} - 1)$

Reset value: 0x0000 0000

This register must *not* be written when the channel is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA [31:16]															
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

Base address of the peripheral data register from/to which the data will be read/written.

When PSIZE is 01 (16-bit), the PA[0] bit is ignored. Access is automatically aligned to a half-word address.

When PSIZE is 10 (32-bit), PA[1:0] are ignored. Access is automatically aligned to a word address.

13.5.6 DMA channel x memory address register (DMA_CMARx) (x = 1..7, where x = channel number)

Address offset: 0x14 + 0d20 × (channel number – 1)

Reset value: 0x0000 0000

This register must *not* be written when the channel is enabled.

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

Bits 31:0 **MA[31:0]**: Memory address

Base address of the memory area from/to which the data will be read/written.

When MSIZE is 01 (16-bit), the MA[0] bit is ignored. Access is automatically aligned to a half-word address.

When MSIZE is 10 (32-bit), MA[1:0] are ignored. Access is automatically aligned to a word address.

13.5.7 DMA register map

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

Table 76. DMA register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DMA_ISR	Res.	Res.			TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	DMA_IFCR	Res.	Res.			CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	DMA_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		MEM2MEM	PL [1:0]	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MNC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN		
	Reset value																	0	0													0	0
0x0C	DMA_CNDTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NDT[15:0]														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	DMA_CPAR1	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	DMA_CMAR1	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	DMA_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		MEM2MEM	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MNC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	Reset value																	0	0												0	0	0
0x20	DMA_CNDTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NDT[15:0]														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	DMA_CPAR2	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	DMA_CMAR2	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	DMA_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		MEM2MEM	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MNC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	Reset value																	0	0												0	0	0
0x34	DMA_CNDTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NDT[15:0]														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	DMA_CPAR3	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	DMA_CMAR3	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	DMA_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		MEM2MEM	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MNC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	Reset value																	0	0												0	0	0
0x48	DMA_CNDTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NDT[15:0]														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 76. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x4C	DMA_CPAR4	PA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x50	DMA_CMAR4	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x54	Reserved																																	
0x58	DMA_CCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN		
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x5C	DMA_CNDTR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x60	DMA_CPAR5	PA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x64	DMA_CMAR5	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x068	Reserved																																	
0x06C	DMA_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN		
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x070	DMA_CNDTR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x074	DMA_CPAR6	PA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x078	DMA_CMAR6	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x07C	Reserved																																	
0x080	DMA_CCR7	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		
0x084	DMA_CNDTR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x088	DMA_CPAR7	PA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x08C	DMA_CMAR7	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x090	Reserved																																	

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

14 Interrupts and events

14.1 Nested vectored interrupt controller (NVIC)

14.1.1 NVIC main features

- 66 maskable interrupt channels (not including the sixteen Cortex®-M4 with FPU interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of System Control Registers

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to the PM0214 programming manual for Cortex®-M4 products.

14.1.2 SysTick calibration value register

The SysTick calibration value is set to 9000, which gives a reference time base of 1 ms with the SysTick clock set to 9 MHz (max $f_{HCLK}/8$).

14.1.3 Interrupt and exception vectors

[Table 77](#) is the vector table for STM32F302xB/C devices. [Table 78](#) is the vector table for STM32F302x6/8 devices.

Table 77. STM32F302xB/C/D/E vector table

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000 0000
	-3	fixed	Reset	Reset	0x0000 0004
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
	-1	fixed	HardFault	All class of fault	0x0000 000C
	0	settable	MemManage	Memory management	0x0000 0010
	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
				Reserved	0x0000 001C - 0x0000 0028

Table 77. STM32F302xB/C/D/E vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
	3	settable	SVCall	System service call via SWI instruction	0x0000 002C
	5	settable	PendSV	Pendable request for system service	0x0000 0038
	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD	PVD through EXTI Line16 detection interrupt	0x0000 0044
2	9	settable	TAMPER_STAMP	Tamper and TimeStamp interrupts through EXTI Line19	0x0000 0048
3	10	settable	RTC_WKUP	RTC wakeup timer interrupt through EXTI Line20	0x0000 004C
4	11	settable	FLASH	Flash global interrupt	0x0000 0050
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2_TS	EXTI Line2 and Touch sensing interrupts	0x0000 0060
9	16	settable	EXTI3	EXTI Line3	0x0000 0064
10	17	settable	EXTI4	EXTI Line4	0x0000 0068
11	18	settable	DMA1_Channel1	DMA1 channel 1 interrupt	0x0000 006C
12	19	settable	DMA1_Channel2	DMA1 channel 2 interrupt	0x0000 0070
13	20	settable	DMA1_Channel3	DMA1 channel 3 interrupt	0x0000 0074
14	21	settable	DMA1_Channel4	DMA1 channel 4 interrupt	0x0000 0078
15	22	settable	DMA1_Channel5	DMA1 channel 5 interrupt	0x0000 007C
16	23	settable	DMA1_Channel6	DMA1 channel 6 interrupt	0x0000 0080
17	24	settable	DMA1_Channel7	DMA1 channel 7 interrupt	0x0000 0084
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000 0088
19 ⁽¹⁾	26	settable	USB_HP/CAN_TX	USB High Priority/CAN_TX interrupts	0x0000 008C
20 ⁽¹⁾	27	settable	USB_LP/CAN_RX0	USB Low Priority/CAN_RX0 interrupts	0x0000 0090
21	28	settable	CAN_RX1	CAN_RX1 interrupt	0x0000 0094
22	29	settable	CAN_SCE	CAN_SCE interrupt	0x0000 0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000 009C
24	31	settable	TIM1_BRK/TIM15	TIM1 Break/TIM15 global interrupts	0x0000 00A0
25	32	settable	TIM1_UP/TIM16	TIM1 Update/TIM16 global interrupts	0x0000 00A4
26	33	settable	TIM1_TRG_COM /TIM17	TIM1 trigger and commutation/TIM17 interrupts	0x0000 00A8
27	34	settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00AC

Table 77. STM32F302xB/C/D/E vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
28	35	settable	TIM2	TIM2 global interrupt	0x0000 00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000 00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000 00B8
31	38	settable	I2C1_EV	I2C1 event interrupt & EXTI Line23 interrupt	0x0000 00BC
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000 00C0
33	40	settable	I2C2_EV	I2C2 event interrupt & EXTI Line24 interrupt	0x0000 00C4
34	41	settable	I2C2_ER	I2C2 error interrupt	0x0000 00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000 00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000 00D0
37	44	settable	USART1	USART1 global interrupt & EXTI Line 25	0x0000 00D4
38	45	settable	USART2	USART2 global interrupt & EXTI Line 26	0x0000 00D8
39	46	settable	USART3	USART3 global interrupt & EXTI Line 28	0x0000 00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
41	48	settable	RTC_Alarm	RTC alarm interrupt	0x0000 00E4
42 ⁽¹⁾	49	settable	USBWakeUp	USB wakeup from Suspend (EXTI line 18)	0x0000 00E8
43	50	settable	Reserved		0x0000 00EC
44	51	settable	Reserved		0x0000 00F0
45	52	settable	Reserved		0x0000 00F4
46	53	settable	Reserved		0x0000 00F8
47	54	settable	Reserved		0x0000 00FC
48	55	settable	FMC ⁽²⁾	FMC global interrupt	0x0000 0100
49	56		Reserved		0x0000 0104
50	57		Reserved		0x0000 0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000 010C
52	59	settable	UART4	UART4 global and EXTI Line 34 interrupts	0x0000 0110
53	60	settable	UART5	UART5 global and EXTI Line 35 interrupts	0x0000 0114
54	61	settable	TIM6_DAC	TIM6 global and DAC1 underrun interrupts.	0x0000 0118
55	62	settable	Reserved		0x0000 011C
56	63	settable	DMA2_Channel1	DMA2 channel1 global interrupt	0x0000 0120
57	64	settable	DMA2_Channel2	DMA2 channel2 global interrupt	0x0000 0124
58	65	settable	DMA2_Channel3	DMA2 channel3 global interrupt	0x0000 0128
59	66	settable	DMA2_Channel4	DMA2 channel4 global interrupt	0x0000 012C
60	67	settable	DMA2_Channel5	DMA2 channel5 global interrupt	0x0000 0130

Table 77. STM32F302xB/C/D/E vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
61	68	settable	Reserved		0x0000 0134
62	69		Reserved		0x0000 0138
63	70		Reserved		0x0000 013C
64	71	settable	COMP1_2_	COMP1 & COMP2 interrupts combined with EXTI Lines 21, 22 interrupts.	0x0000 0140
65	72	settable	COMP4_6	COMP4 & COMP6 interrupts combined with EXTI Lines 30 and 32 interrupts.	0x0000 0144
66	73	settable	Reserved		0x0000 0148
67	74		Reserved		0x0000 014C
68	75		Reserved		0x0000 0150
69	76		Reserved		0x0000 0154
70	77		Reserved		0x0000 0158
71	78		Reserved		0x0000 015C
72	79	settable	I2C3_EV ⁽²⁾	I2C3 Event interrupt	0x0000 0160
73	80	settable	I2C3_ER ⁽²⁾	I2C3 Error interrupt	0x0000 0164
74	81	settable	USB_HP	USB High priority interrupt	0x0000 0168
75	82	settable	USB_LP	USB Low priority interrupt	0x0000 016C
76	83	settable	USB_WakeUp_RMP (see note 1)	USB wake up from Suspend and EXTI Line 18	0x0000 0170
81	88	settable	FPU	Floating point interrupt	0x0000 0184
84	91	settable		SPI4 SPI4 Global interrupt ⁽²⁾	0x0000 0190

1. It is possible to remap the USB interrupts (USB_HP, USB_LP and USB_WKUP) on interrupt lines 74, 75 and 76 respectively by setting the USB_IT_RMP bit in the [Section 12.1.1: SYSCFG configuration register 1 \(SYSCFG_CFGR1\) on page 233](#).

2. Available in STM32F302xD/E only.

Table 78. STM32F302x6/8 vector table

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000 0000
	-3	fixed	Reset	Reset	0x0000 0004
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
	-1	fixed	HardFault	All class of fault	0x0000 000C

Table 78. STM32F302x6/8 vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
	0	settable	MemManage	Memory management	0x0000 0010
	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
				Reserved	0x0000 001C - 0x0000 0028
	3	settable	SVCall	System service call via SWI instruction	0x0000 002C
	5	settable	PendSV	Pendable request for system service	0x0000 0038
	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD	PVD through EXTI line 16 detection interrupt	0x0000 0044
2	9	settable	TAMPER_STAMP	Tamper and TimeStamp interrupts through the EXTI line 19	0x0000 0048
3	10	settable	RTC_WKUP	RTC wakeup timer interrupts through the EXTI line 20	0x0000 004C
4	11	settable	FLASH	Flash global interrupt	0x0000 0050
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2_TS	EXTI Line2 and Touch sensing interrupts	0x0000 0060
9	16	settable	EXTI3	EXTI Line3	0x0000 0064
10	17	settable	EXTI4	EXTI Line4	0x0000 0068
11	18	settable	DMA1_Channel1	DMA1 channel 1 interrupt	0x0000 006C
12	19	settable	DMA1_Channel2	DMA1 channel 2 interrupt	0x0000 0070
13	20	settable	DMA1_Channel3	DMA1 channel 3 interrupt	0x0000 0074
14	21	settable	DMA1_Channel4	DMA1 channel 4 interrupt	0x0000 0078
15	22	settable	DMA1_Channel5	DMA1 channel 5 interrupt	0x0000 007C
16	23	settable	DMA1_Channel6	DMA1 channel 6 interrupt	0x0000 0080
17	24	settable	DMA1_Channel7	DMA1 channel 7 interrupt	0x0000 0084
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000 0088
19	26	settable	CAN_TX	CAN_TX interrupts	0x0000 008C
20	27	settable	CAN_RX0	CAN_RX0 interrupts	0x0000 0090
21	28	settable	CAN_RX1	CAN_RX1 interrupt	0x0000 0094
22	29	settable	CAN_SCE	CAN_SCE interrupt	0x0000 0098

Table 78. STM32F302x6/8 vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000 009C
24	31	settable	TIM1_BRK/TIM15	TIM1 Break/TIM15 global interrupts	0x0000 00A0
25	32	settable	TIM1_UP/TIM16	TIM1 Update/TIM16 global interrupts	0x0000 00A4
26	33	settable	TIM1_TRG_COM /TIM17	TIM1 trigger and commutation/TIM17 interrupts	0x0000 00A8
27	34	settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000 00B0
29	36		Reserved		0x0000 00B4
30	37		Reserved		0x0000 00B8
31	38	settable	I2C1_EV	I2C1 event interrupt & EXTI Line23 interrupt	0x0000 00BC
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000 00C0
33	40		I2C2_EV	I2C2 event interrupt	0x0000 00C4
34	41		I2C2_ER	I2C2 error interrupt	0x0000 00C8
35	42		Reserved		0x0000 00CC
36	43		SPI2	SPI2 global interrupt	0x0000 00D0
37	44	settable	USART1	USART1 global interrupt & EXTI Line 25	0x0000 00D4
38	45	settable	USART2	USART2 global interrupt	0x0000 00D8
39	46	settable	USART3	USART3 global interrupt	0x0000 00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
41	48	settable	RTC_Alarm	RTC alarm interrupt	0x0000 00E4
42	49		USBWakeUp	USB Wakeup interrupt	0x0000 00E8
43	50		Reserved		0x0000 00EC
44	51		Reserved		0x0000 00F0
45	52		Reserved		0x0000 00F4
46	53		Reserved		0x0000 00F8
47	54		Reserved		0x0000 00FC
48	55		Reserved		0x0000 0100
49	56		Reserved		0x0000 0104
50	57		Reserved		0x0000 0108
51	58		SPI3	SPI3 global interrupt	0x0000 010C
52	59		Reserved		0x0000 0110
53	60		Reserved		0x0000 0114
54	61	settable	TIM6_DAC	TIM6 global and DAC1 underrun interrupts	0x0000 0118

Table 78. STM32F302x6/8 vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
55	62		Reserved		0x0000 011C
56	63		Reserved		0x0000 0120
57	64		Reserved		0x0000 0124
58	65		Reserved		0x0000 0128
59	66		Reserved		0x0000 012C
60	67		Reserved		0x0000 0130
61	68		Reserved		0x0000 0134
62	69		Reserved		0x0000 0138
63	70		Reserved		0x0000 013C
64	71	settable	COMP2	COMP2 interrupt combined with EXTI Lines 22 interrupt.	0x0000 0140
65	72	settable	COMP4_6	COMP4 & COMP6 interrupts combined with EXTI Lines 30 and 32 interrupts respectively.	0x0000 0144
67	74		Reserved		0x0000 014C
68	75		Reserved		0x0000 0150
69	76		Reserved		0x0000 0154
70	77		Reserved		0x0000 0158
71	78		Reserved		0x0000 015C
72	79		I2C3_EV	I2C3 event interrupt & EXTI Line27 interrupt	0x0000 0160
73	80		I2C3_ER	I2C3 error interrupt	0x0000 0164
74	81		USB_HP	USB High Priority global interrupt remap	0x0000 0168
75	82		USB_LP	USB Low Priority global interrupt remap	0x0000 016C
76	83		USBWakeUp_RMP	USB Wakeup interrupt remap	0x0000 0170
77	84		Reserved		0x0000 0174
78	85		Reserved		0x0000 0178
79	86		Reserved		0x0000 017C
80	87		Reserved		0x0000 0180
81	88	settable	FPU	Floating point interrupt	0x0000 0184

14.2 Extended interrupts and events controller (EXTI)

The extended interrupts and events controller (EXTI) manages the external and internal asynchronous events/interrupts and generates the event request to the CPU/Interrupt Controller and a wake-up request to the Power Manager.

The EXTI allows the management of up to 36 external/internal event line (28 external event lines and 8 internal event lines).

The active edge of each external interrupt line can be chosen independently, whilst for internal interrupt the active edge is always the rising one. An interrupt could be left pending: in case of an external one, a status register is instantiated and indicates the source of the interrupt; an event is always a simple pulse and it's used for triggering the core wake-up. For internal interrupts, the pending status is assured by the generating peripheral, so no need for a specific flag. Each input line can be masked independently for interrupt or event generation, in addition the internal lines are sampled only in STOP mode. This controller allows also to emulate the (only) external events by software, multiplexed with the corresponding hardware event line, by writing to a dedicated register.

14.2.1 Main features

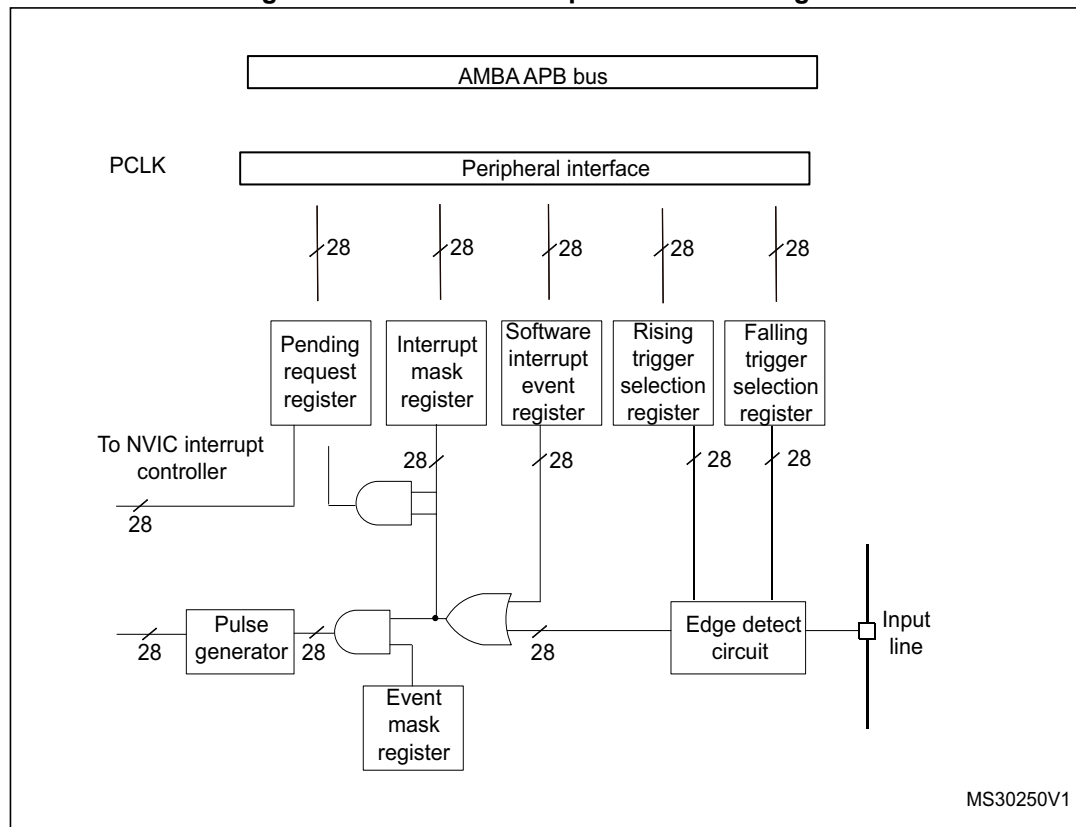
The EXTI main features are the following:

- support generation of up to 36 event/interrupt requests
- Independent configuration of each line as an external or an internal event requests
- Independent mask on each event/interrupt line
- Automatic disable of internal lines when system is not in STOP mode
- Independent trigger for external event/interrupt line
- Dedicated status bit for external interrupt line
- Emulation for all the external event requests.

14.2.2 Block diagram

The extended interrupt/event block diagram is shown in the following figure.

Figure 49. External interrupt/event block diagram



14.2.3 Wakeup event management

STM32F302xx devices are able to handle external or internal events in order to wake up the core (WFE). The wakeup event can be generated either by:

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex[®]-M4 System Control register. When the MCU resumes from WFE, the EXTI peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- or by configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

14.2.4 Asynchronous Internal Interrupts

Some communication peripherals (UART, I2C) are able to generate events when the system is in run mode and also when the system is in stop mode allowing to wake up the system from stop mode.

To accomplish this, the peripheral is asked to generate both a synchronized (to the system clock, e.g. APB clock) and an asynchronous version of the event.

14.2.5 Functional description

For the external interrupt lines, to generate the interrupt, the interrupt line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a '1' to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is reset by writing a '1' in the pending register.

For the internal interrupt lines, the active edge is always the rising edge, the interrupt is enabled by default in the interrupt mask register and there is no corresponding pending bit in the pending register.

To generate the event, the event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the event request by writing a '1' to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event pulse is generated. The pending bit corresponding to the event line is not set.

For the external lines, an interrupt/event request can also be generated by software by writing a '1' in the software interrupt/event register.

Note: The interrupts or events associated to the internal lines can be triggered only when the system is in STOP mode. If the system is still running, no interrupt/event is generated.

Hardware interrupt selection

To configure a line as interrupt source, use the following procedure:

- Configure the corresponding mask bit in the EXTI_IMR register.
- Configure the Trigger Selection bits of the Interrupt line (EXTI_RTSR and EXTI_FTSR)
- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the EXTI so that an interrupt coming from one of the EXTI line can be correctly acknowledged.

Hardware event selection

To configure a line as event source, use the following procedure:

- Configure the corresponding mask bit in the EXTI_EMR register.
- Configure the Trigger Selection bits of the Event line (EXTI_RTSR and EXTI_FTSR)

Software interrupt/event selection

Any of the external lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt.

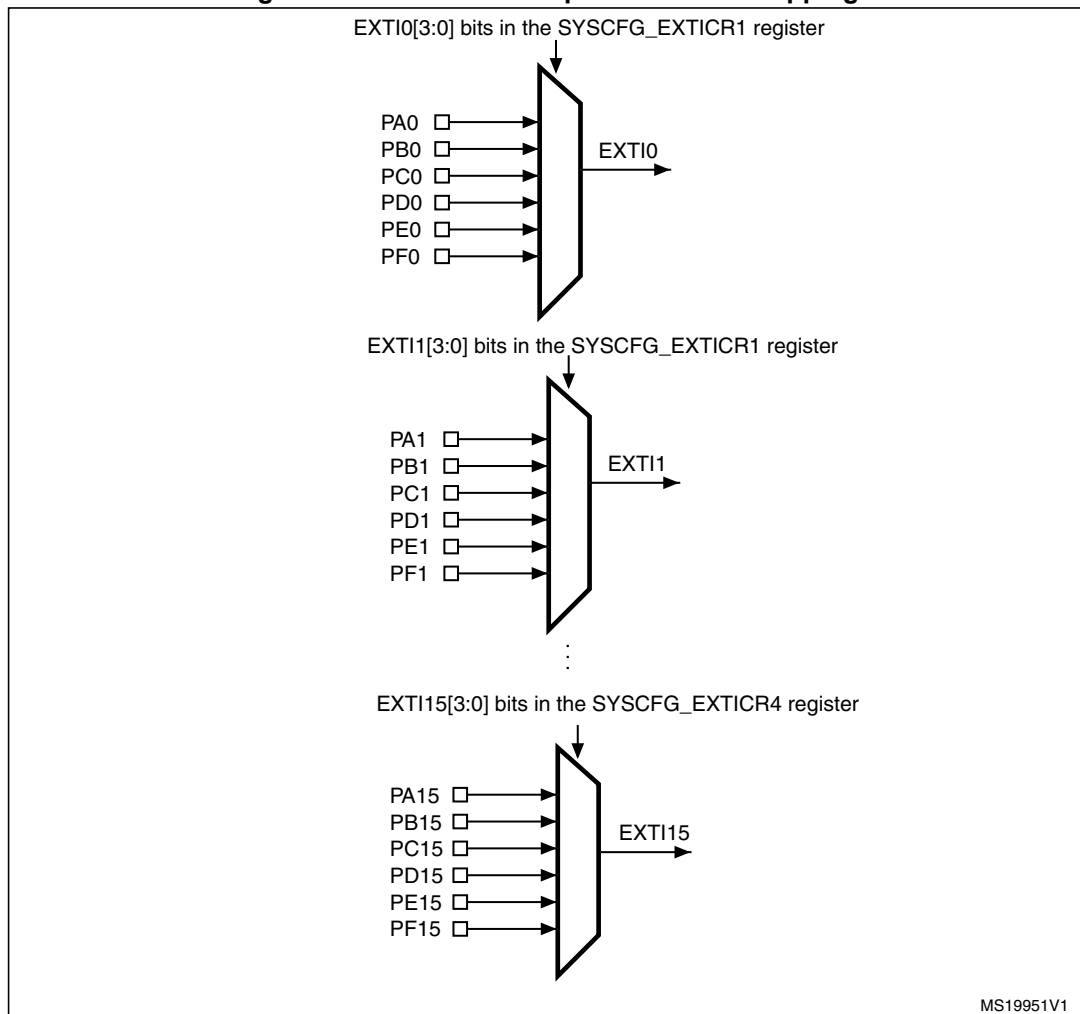
- Configure the corresponding mask bit (EXTI_IMR, EXTI_EMR)
- Set the required bit of the software interrupt register (EXTI_SWIER)

14.2.6 External and internal interrupt/event line mapping

36 interrupt/event lines are available: 8 lines are internal (including the reserved ones); the remaining 28 lines are external.

The GPIOs are connected to the 16 external interrupt/event lines in the following manner:

Figure 50. External interrupt/event GPIO mapping



The remaining lines are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event
- EXTI line 18 is connected to USB Device FS wakeup event
- EXTI line 19 is connected to RTC tamper and Timestamps
- EXTI line 20 is connected to RTC wakeup timer
- EXTI line 21 is connected to Comparator 1 output (STM32F302xB/C/D/E and STM32F302x6/8 devices)
- EXTI line 22 is connected to Comparator 2 output
- EXTI line 23 is connected to I2C1 wakeup
- EXTI line 24 is connected to I2C2 wakeup
- EXTI line 25 is connected to USART1 wakeup
- EXTI line 26 is connected to USART2 wakeup
- EXTI line 27 is connected to I2C3 wakeup (STM32F302x6/8 devices only)
- EXTI line 28 is connected to USART3 wakeup
- EXTI line 29 is reserved
- EXTI line 30 is connected to Comparator 4 output
- EXTI line 31 is reserved
- EXTI line 32 is connected to Comparator 6 output
- EXTI line 33 is reserved
- EXTI line 34 is connected to UART4 wakeup (STM32F302xB/C/D/E and STM32F302x6/8 devices)
- EXTI line 35 is connected to UART5 wakeup (STM32F302xB/C/D/E and STM32F302x6/8 devices)

Note: EXTI lines 29, 31 and 33 are internal.

14.3 EXTI registers

Refer to [Section 2.1 on page 42](#) for a list of abbreviations used in register descriptions.

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

14.3.1 Interrupt mask register (EXTI_IMR1)

Address offset: 0x00

Reset value: 0x1F80 0000 (See note below)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR31	MR30	MR29	MR28	MR27	MR26	MR25	MR24	MR23	MR22	MR21	MR20	MR19	MR18	MR17	MR16
r/w	r/w	r/w	r/w	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
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
r/w	r/w	r/w	r/w	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 **MRx**: Interrupt Mask on external/internal line x

0: Interrupt request from Line x is masked

1: Interrupt request from Line x is not masked

Note: The reset value for the internal lines (23, 24, 25, 26, 27 and 28) is set to '1' in order to enable the interrupt by default.

14.3.2 Event mask register (EXTI_EMR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR31	MR30	MR29	MR28	MR27	MR26	MR25	MR24	MR23	MR22	MR21	MR20	MR19	MR18	MR17	MR16
r/w	r/w	r/w	r/w	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
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
r/w	r/w	r/w	r/w	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 **MRx**: Event mask on external/internal line x

0: Event request from Line x is masked

1: Event request from Line x is not masked

14.3.3 Rising trigger selection register (EXTI_RTSTR1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TR31	TR30	TR29	Res.	Res.	Res.	Res.	Res.	Res.	TR22	TR21	TR20	TR19	TR18	TR17	TR16
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
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TRx**: Rising trigger event configuration bit of line x (x = 31 to 29)

0: Rising trigger disabled (for Event and Interrupt) for input line

1: Rising trigger enabled (for Event and Interrupt) for input line.

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **TRx**: Rising trigger event configuration bit of line x (x = 22 to 0)

0: Rising trigger disabled (for Event and Interrupt) for input line

1: Rising trigger enabled (for Event and Interrupt) for input line.

Note: *The external wakeup lines are edge-triggered. No glitches must be generated on these lines. If a rising edge on an external interrupt line occurs during a write operation in the EXTI_RTSTR register, the pending bit is not set.*

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

14.3.4 Falling trigger selection register (EXTI_FTSR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TR31	TR30	TR29	Res.	Res.	Res.	Res.	Res.	Res.	TR22	TR21	TR20	TR19	TR18	TR17	TR16
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
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TRx**: Falling trigger event configuration bit of line x (x = 31 to 29)

0: Falling trigger disabled (for Event and Interrupt) for input line

1: Falling trigger enabled (for Event and Interrupt) for input line.

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **TRx**: Falling trigger event configuration bit of line x (x = 22 to 0)

0: Falling trigger disabled (for Event and Interrupt) for input line

1: Falling trigger enabled (for Event and Interrupt) for input line.

Note: The external wakeup lines are edge-triggered. No glitches must be generated on these lines. If a falling edge on an external interrupt line occurs during a write operation to the EXTI_FTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

14.3.5 Software interrupt event register (EXTI_SWIER1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWIER 31	SWIER 30	SWIER 29	Res.	Res.	Res.	Res.	Res.	Res.	SWIER 22	SWIER 21	SWIER 20	SWIER 19	SWIER 18	SWIER 17	SWIER 16
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
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
r/w	r/w	r/w	r/w	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: 29 **SWIERx**: Software interrupt on line x (x = 31 o 29)

If the interrupt is enabled on this line in the EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit in the EXTI_PR register (by writing a '1' into the bit).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **SWIERx**: Software interrupt on line x (x = 22 to 0)

If the interrupt is enabled on this line in the EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit of EXTI_PR (by writing a '1' into the bit).

14.3.6 Pending register (EXTI_PR1)

Address offset: 0x14

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PR31	PR30	PR29	Res.	Res.	Res.	Res.	Res.	Res.	PR22	PR21	PR20	PR19	PR18	PR17	PR16
rc_w1	rc_w1	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
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:29 **PRx**: Pending bit on line x (x = 31 to 29)

0: No trigger request occurred

1: Selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line.

This bit is cleared by writing a '1' to the bit.

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **PRx**: Pending bit on line x (x = 22 to 0)

0: No trigger request occurred

1: Selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line.

This bit is cleared by writing a '1' to the bit.

14.3.7 Interrupt mask register (EXTI_IMR2)

Address offset: 0x20

Reset value: 0xFFFF FFC (See note below)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	MR35	MR34	MR33	MR32
												rw	rw	rw	rw

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

Bits 3:0 **MRx**: Interrupt Mask on external/internal line x; x = 32..35

0: Interrupt request from Line x is masked

1: Interrupt request from Line x is not masked

Note: The reset value for the internal lines (EXTI Lines 34 and 35) and the reserved lines is set to '1'.

14.3.8 Event mask register (EXTI_EMR2)

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.	Res.	Res.	MR35	MR34	MR33	MR32
												rw	rw	rw	rw

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

Bits 3:0 **MRx**: Event mask on external/internal line x, x = 32..35

0: Event request from Line x is masked

1: Event request from Line x is not masked

14.3.9 Rising trigger selection register (EXTI_RTSR2)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR33	TR32
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TRx**: Rising trigger event configuration bit of line x (x = 32, 33)

0: Rising trigger disabled (for Event and Interrupt) for input line

1: Rising trigger enabled (for Event and Interrupt) for input line.

Note: *The external wakeup lines are edge-triggered. No glitches must be generated on these lines. If a rising edge on an external interrupt line occurs during a write operation to the EXTI_RTSR register, the pending bit is not set.*

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

14.3.10 Falling trigger selection register (EXTI_FTSR2)

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.	Res.	Res.	TR33	TR32
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TRx**: Falling trigger event configuration bit of line x (x = 32,33)

0: Falling trigger disabled (for Event and Interrupt) for input line

1: Falling trigger enabled (for Event and Interrupt) for input line.

Note: The external wakeup lines are edge-triggered. No glitches must be generated on these lines. If a falling edge on an external interrupt line occurs during a write operation to the EXTI_FTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

14.3.11 Software interrupt event register (EXTI_SWIER2)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER 33	SWIER 32
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **SWIERx**: Software interrupt on line x (x = 32, 33)

If the interrupt is enabled on this line in the EXTI_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit of EXTI_PR (by writing a '1' to the bit).

14.3.12 Pending register (EXTI_PR2)

Address offset: 0x34

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	PR33	PR32
														rc_w1	rc_w1

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **PRx**: Pending bit on line x (x = 32,33)

0: No trigger request occurred

1: Selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line.

This bit is cleared by writing a '1' into the bit.

14.3.13 EXTI register map

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

Table 79. External interrupt/event controller register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	EXTI_IMR1	MR[31:0]																																
	Reset value	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	EXTI_EMR1	MR[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	EXTI_RTISR1	TR[31:29]		Res.		Res.		Res.		Res.		Res.		TR[22:0]																				
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	
0x0C	EXTI_FTISR1	TR[31:29]		Res.		Res.		Res.		Res.		Res.		TR[22:0]																				
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	
0x10	EXTI_SWIER1	SWIER[31:29]		Res.		Res.		Res.		Res.		Res.		SWIER[22:0]																				
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	
0x14	EXTI_PR1	PR[31:29]		Res.		Res.		Res.		Res.		Res.		PR[22:0]																				
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	
0x20	EXTI_IMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR35	MR34	MR33	MR32	
	Reset value																												1	1	0	0		
0x24	EXTI_EMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR35	MR34	MR33	MR32	
	Reset value																												0	0	0	0		
0x28	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.	TR33	TR32	TR31	TR30
	Reset value																													0	0	0	0	
0x2C	EXTI_FTISR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR33	TR32	TR31	TR30
	Reset value																													0	0	0	0	

Table 79. External interrupt/event controller register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	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.	Res.	Res.
	Reset value																																
0x34	EXTI_PR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

15 Analog-to-digital converters (ADC)

15.1 Introduction

This section describes the implementation of up to 2 ADCs:

ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).

Each ADC consists of a 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 19 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

Note: The STM32F302x6/8 devices have only ADC1. The STM32F302xB/C/D/E have ADC1 and ADC2.

15.2 ADC main features

- High-performance features
 - STM32F302xB/C/D/E devices have two ADCs which can operate in dual mode.
 - STM32F302x6/x8 devices have one ADC

The table below summarizes the different external channels available per ADC.

Table 80. ADC external channels mapping

Device	ADC1	ADC2
STM32F302x6/8	15	N.A
STM32F302xB/C	10	12
STM32F302xD/E	11	13

- 12, 10, 8 or 6-bit configurable resolution
- ADC conversion time:
 - Fast channels: 0.19 μ s for 12-bit resolution (5.1 Ms/s)
 - Slow channels: 0.21 μ s for 12-bit resolution (4.8 Ms/s)
- ADC conversion time is independent from the AHB bus clock frequency
- Faster conversion time by lowering resolution: 0.16 μ s for 10-bit resolution
- Can manage Single-ended or differential inputs (programmable per channels)
- AHB slave bus interface to allow fast data handling
- Self-calibration
- Channel-wise programmable sampling time
- Up to four injected channels (analog inputs assignment to regular or injected channels is fully configurable)
- Hardware assistant to prepare the context of the injected channels to allow fast context switching
- Data alignment with in-built data coherency
- Data can be managed by GP-DMA for regular channel conversions
- 4 dedicated data registers for the injected channels
- Low-power features
 - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
 - Allows slow bus frequency application while keeping optimum ADC performance (0.19 μ s conversion time for fast channels can be kept whatever the AHB bus clock frequency)
 - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (auto-delayed mode)
- In addition, there are internal dedicated channels available per ADC. See the table below:

Table 81. ADC internal channels summary

Product	ADC1	ADC2	Total of internal ADC channels
STM32F302xB/C/D/E	<ul style="list-style-type: none"> – 1 channel connected to temperature sensor. – 1 channel connected to VBAT/2 – 1 channel connected to VREFINT – 1 channel connected to OPAMP1 reference voltage output (VREFOPAMP1). 	<ul style="list-style-type: none"> – 1 channel connected to temperature sensor. – 1 channel connected to VBAT/2 – 1 channel connected to VREFINT. – 1 channel connected to OPAMP2 reference voltage output (VREFOPAMP2). 	5
STM32F302x6/8	<ul style="list-style-type: none"> – 1 channel connected to temperature sensor. – 1 channel connected to VBAT/2 – 1 channel connected to VREFINT 	-	3

- Start-of-conversion can be initiated:
 - by software for both regular and injected conversions
 - by hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions
- Conversion modes
 - Each ADC can convert a single channel or can scan a sequence of channels
 - Single mode converts selected inputs once per trigger
 - Continuous mode converts selected inputs continuously
 - Discontinuous mode
- Dual ADC mode (STM32F302xB/C only)
- Interrupt generation at the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC
- ADC supply requirements: 1.80 V to 3.6 V
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$

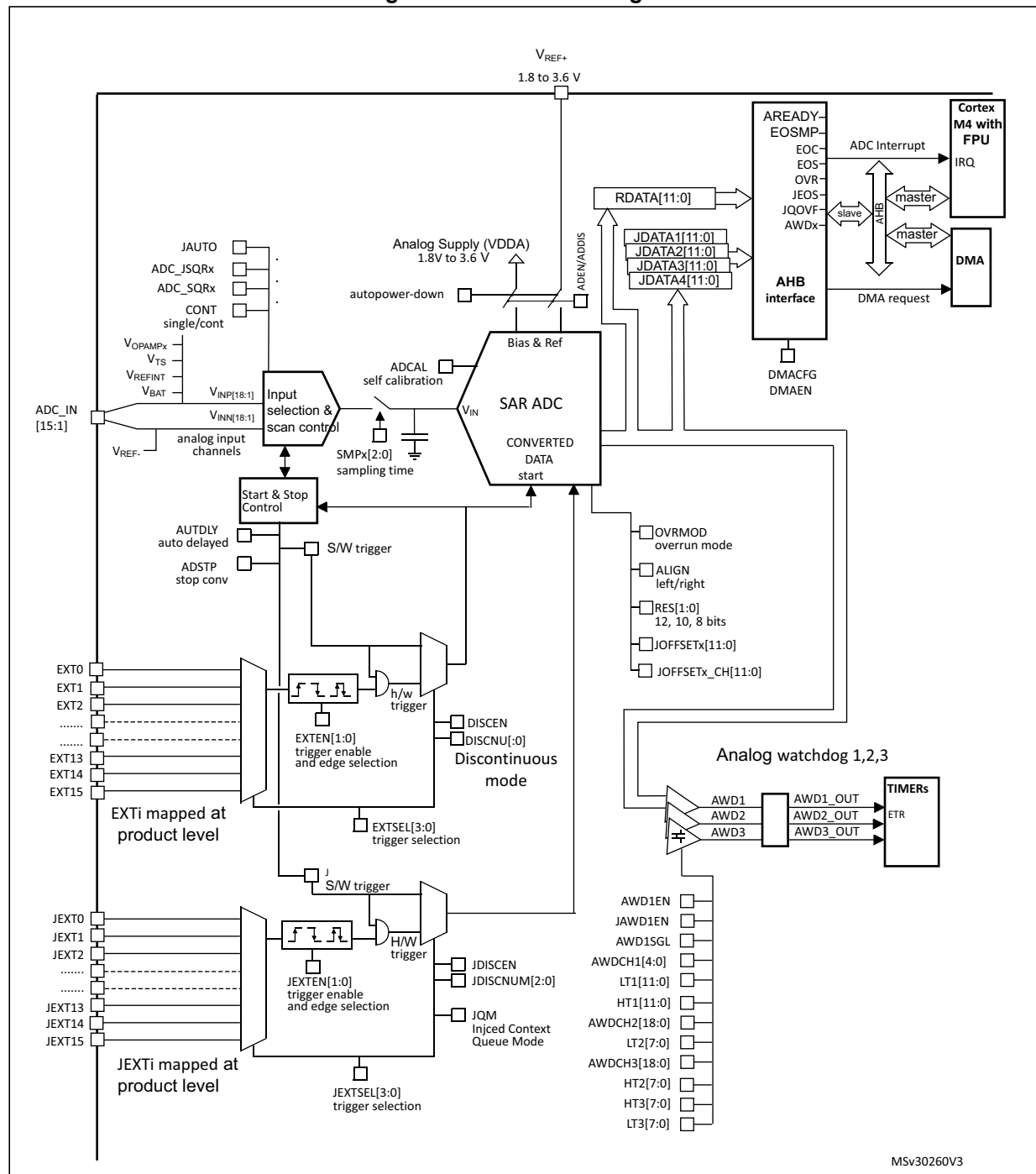
Figure 51 shows the block diagram of one ADC.

15.3 ADC functional description

15.3.1 ADC block diagram

Figure 51 shows the ADC block diagram and Table 83 gives the ADC pin description.

Figure 51. ADC block diagram



MSv30260V3

15.3.2 Pins and internal signals

Table 82. ADC internal signals

Internal signal name	Signal type	Description
EXT[15:0]	Inputs	Up to 16 external trigger inputs for the regular conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.
JEXT[15:0]	Inputs	Up to 16 external trigger inputs for the injected conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.
ADC1_AWDx_OUT ADC2_AWDx_OUT	Output	Internal analog watchdog output signal connected to on-chip timers. (x = Analog watchdog number 1,2,3)
V _{REFOPAMP1}	Input	Reference voltage output from internal operational amplifier 1
V _{REFOPAMP2}	Input	Reference voltage output from internal operational amplifier 2
V _{TS}	Input	Output voltage from internal temperature sensor
V _{REFINT}	Input	Output voltage from internal reference voltage
V _{BAT}	Input supply	External battery voltage supply

Table 83. ADC pins

Name	Signal type	Comments
V _{REF+}	Input, analog reference positive	The higher/positive reference voltage for the ADC, $1.8\text{ V} \leq V_{\text{REF+}} \leq V_{\text{DDA}}$
V _{DDA}	Input, analog supply	Analog power supply equal V _{DDA} : $1.8\text{ V} \leq V_{\text{DDA}} \leq 3.6\text{ V}$
V _{REF-}	Input, analog reference negative	The lower/negative reference voltage for the ADC, V _{REF-} = V _{SSA}
V _{SSA}	Input, analog supply ground	Ground for analog power supply equal to V _{SS}
V _{INP} [18:1]	Positive input analog channels for each ADC	Connected either to external channels: ADC_IN <i>i</i> or internal channels.
V _{INN} [18:1]	Negative input analog channels for each ADC	Connected to V _{REF-} or external channels: ADC_IN <i>i-1</i>
ADCx_IN15:1	External analog input signals	Up to 16 analog input channels (x = ADC number = 1 or 2): – 5 fast channels – 10 slow channels

15.3.3 Clocks

Dual clock domain architecture

The dual clock-domain architecture means that each ADC clock is independent from the AHB bus clock.

The input clock of the two ADCs (master and slave) can be selected between two different clock sources (see [Figure 52: ADC clock scheme](#)):

- a) The ADC clock can be a specific clock source, named “ADCxy_CK (xy=12 or 34) which is independent and asynchronous with the AHB clock”.

It can be configured in the RCC to deliver up to 72 MHz (PLL output). Refer to RCC Section for more information on generating ADC12_CK.

To select this scheme, bits CKMODE[1:0] of the ADCx_CCR register must be reset.

- b) The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). In this mode, a programmable divider factor can be selected (/1, 2 or 4 according to bits CKMODE[1:0]).

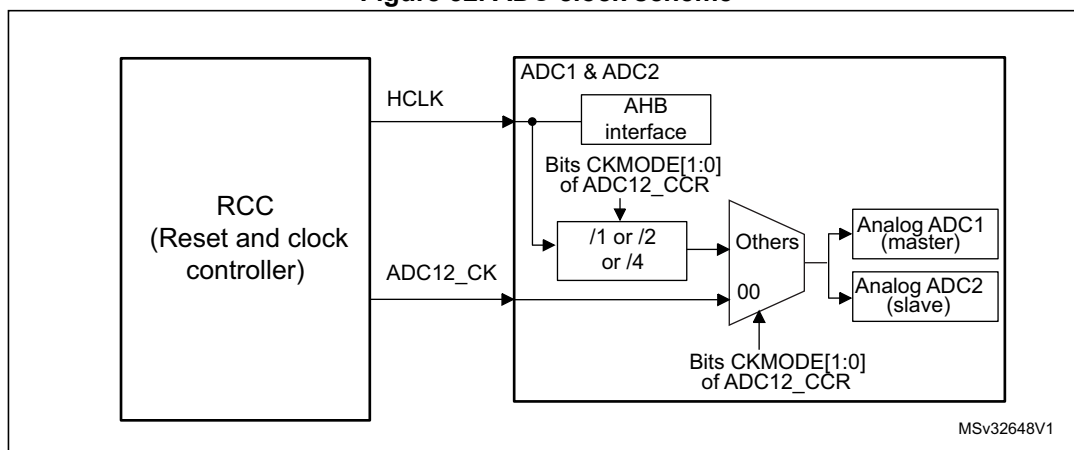
To select this scheme, bits CKMODE[1:0] of the ADCx_CCR register must be different from “00”.

Note: Software can use option a) by writing CKMODE[1:0]=01 or 10 only if the AHB prescaler of the RCC is set to 1 (the duty cycle of the AHB clock must be 50% in this configuration).

Option a) has the advantage of reaching the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, 256; using the prescaler configured with bits ADCxPRES[4:0] in register RCC_CFGR2 (Refer to [Section 10: Reset and clock control \(RCC\)](#)).

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).

Figure 52. ADC clock scheme



1. Refer to the RCC section to see how HCLK and ADC12_CK can be generated.

Clock ratio constraint between ADC clock and AHB clock

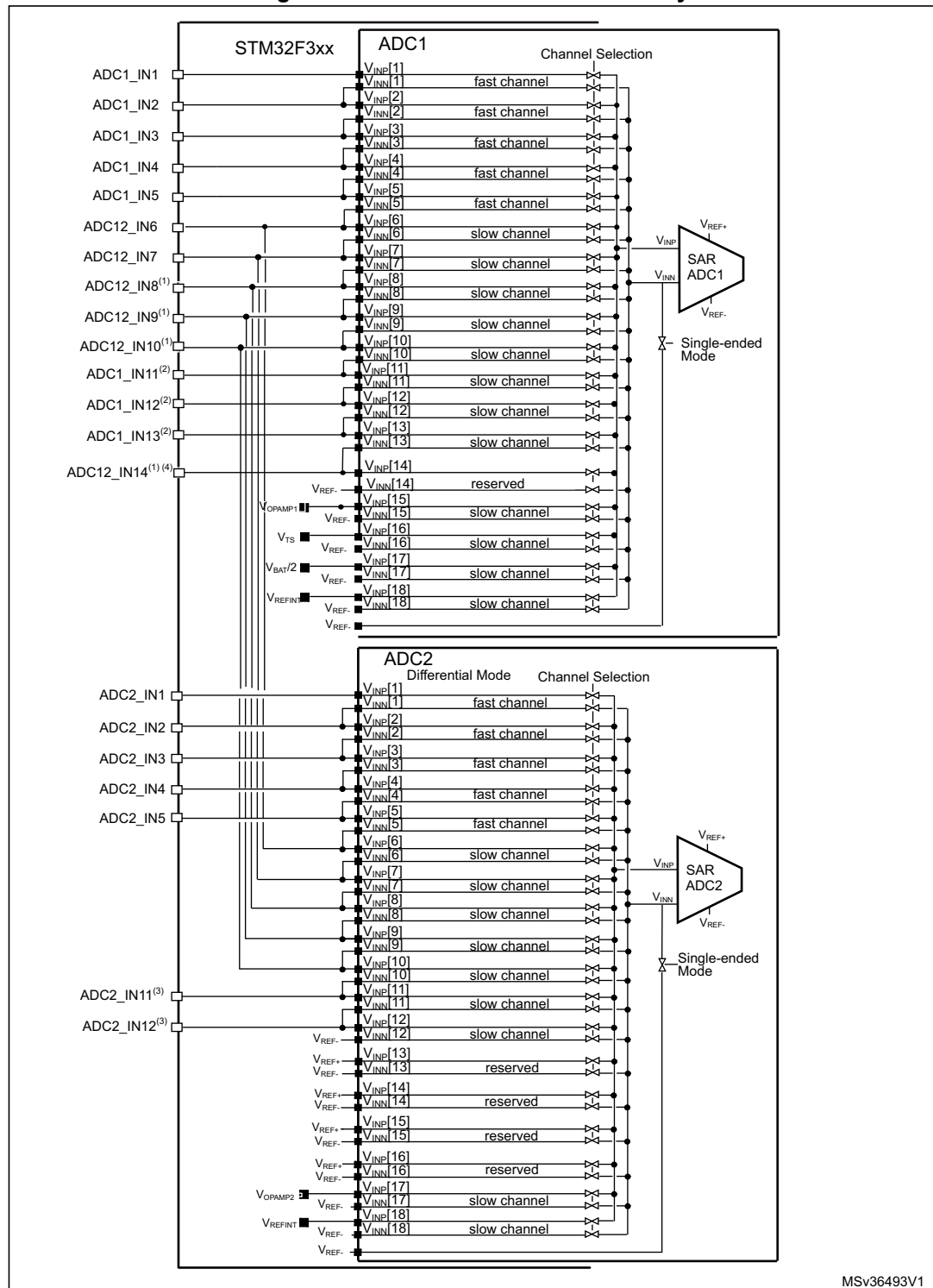
There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{HCLK} \geq F_{ADC} / 4$ if the resolution of all channels are 12-bit or 10-bit
- $F_{HCLK} \geq F_{ADC} / 3$ if there are some channels with resolutions equal to 8-bit (and none with lower resolutions)
- $F_{HCLK} \geq F_{ADC} / 2$ if there are some channels with resolutions equal to 6-bit

15.3.4 ADC1/2 connectivity

ADC1 and ADC2 are tightly coupled and share some external channels as described in [Figure 53](#).

Figure 53. ADC1 and ADC2 connectivity



1. On STM32F302x6/8, this channel is available on ADC1 only.
2. STM32F302x6/8 devices only.
3. On STM32F302xB/CD/E devices only.
4. On STM32F302x6/8/D/E devices.

15.3.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

15.3.6 ADC voltage regulator (ADVREGEN)

The sequence below is required to start ADC operations:

1. Enable the ADC internal voltage regulator (refer to the ADC voltage regulator enable sequence).
2. The software must wait for the startup time of the ADC voltage regulator ($T_{\text{ADCVREG_STUP}}$) before launching a calibration or enabling the ADC. This temporization must be implemented by software. $T_{\text{ADCVREG_STUP}}$ is equal to 10 μs in the worst case process/temperature/power supply.

After ADC operations are complete, the ADC is disabled ($\text{ADEN}=0$).

It is possible to save power by disabling the ADC voltage regulator (refer to the ADC voltage regulator disable sequence).

Note: When the internal voltage regulator is disabled, the internal analog calibration is kept.

ADVREG enable sequence

To enable the ADC voltage regulator, perform the sequence below:

1. Change $\text{ADVREGEN}[1:0]$ bits from '10' (disabled state, reset state) into '00'.
2. Change $\text{ADVREGEN}[1:0]$ bits from '00' into '01' (enabled state).

ADVREG disable sequence

To disable the ADC voltage regulator, perform the sequence below:

1. Change $\text{ADVREGEN}[1:0]$ bits from '01' (enabled state) into '00'.
2. Change $\text{ADVREGEN}[1:0]$ bits from '00' into '10' (disabled state)

15.3.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by writing into bits $\text{DIFSEL}[15:1]$ in the ADCx_DIFSEL register. This configuration must be written while the ADC is disabled ($\text{ADEN}=0$). Note that $\text{DIFSEL}[18:16]$ are fixed to single ended channels (internal channels only) and are always read as 0.

In single-ended input mode, the analog voltage to be converted for channel "i" is the difference between the external voltage ADC_IN_i (positive input) and $V_{\text{REF-}}$ (negative input).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage ADC_IN*i* (positive input) and ADC_IN*i*+1 (negative input).

For a complete description of how the input channels are connected for each ADC, refer to [Figure 53: ADC1 and ADC2 connectivity on page 293](#).

Caution: When configuring the channel “i” in differential input mode, its negative input voltage is connected to ADC_IN*i*+1. As a consequence, channel “i+1” is no longer usable in single-ended mode or in differential mode and must never be configured to be converted. Some channels are shared between ADC1 and ADC2: this can make the channel on the other ADC unusable. Only exception is interleave mode for ADC master and the slave.

Example: Configuring ADC1_IN5 in differential input mode will make ADC12_IN6 not usable: in that case, the channels 6 of both ADC1 and ADC2 must never be converted.

Note: Channels 16, 17 and 18 of ADC1 and channels 17 and 18 of ADC2 are connected to internal analog channels and are internally fixed to single-ended inputs configuration (corresponding bits DIFSEL[i] is always zero). Channel 15 of ADC1 is also an internal channel and the user must configure the corresponding bit DIFSEL[15] to zero.

15.3.8 Calibration (ADCAL, ADCALDIF, ADCx_CALFACT)

Each ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 7-bit wide and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF=0 before launching a calibration which will be applied for single-ended input conversions.
- Write ADCALDIF=1 before launching a calibration which will be applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL=1. Calibration can only be initiated when the ADC is disabled (when ADEN=0). ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of ADCx_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if the ADC is disabled (ADEN=0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

The internal analog calibration is 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, V_{DDA} and temperature change to a lesser extent), it is recommended to re-run a calibration cycle.

The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in STANDBY or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the

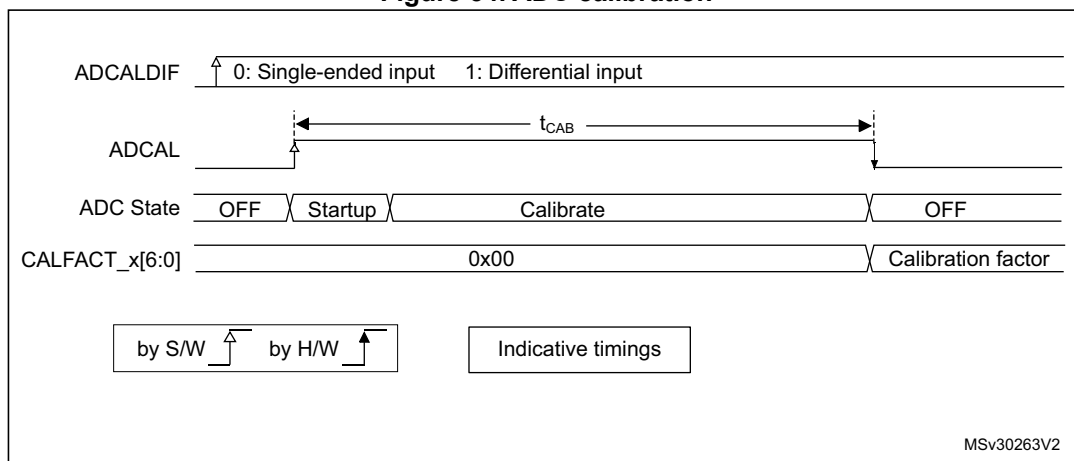
ADCx_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN=1 and ADSTART=0 and JADSTART=0). Then, at the next start of conversion, the calibration factor will automatically be injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion.

Software procedure to calibrate the ADC

1. Ensure ADVREGEN[1:0]=01 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN=0.
3. Select the input mode for this calibration by setting ADCALDIF=0 (Single-ended input) or ADCALDIF=1 (Differential input).
4. Set ADCAL=1.
5. Wait until ADCAL=0.
6. The calibration factor can be read from ADCx_CALFACT register.

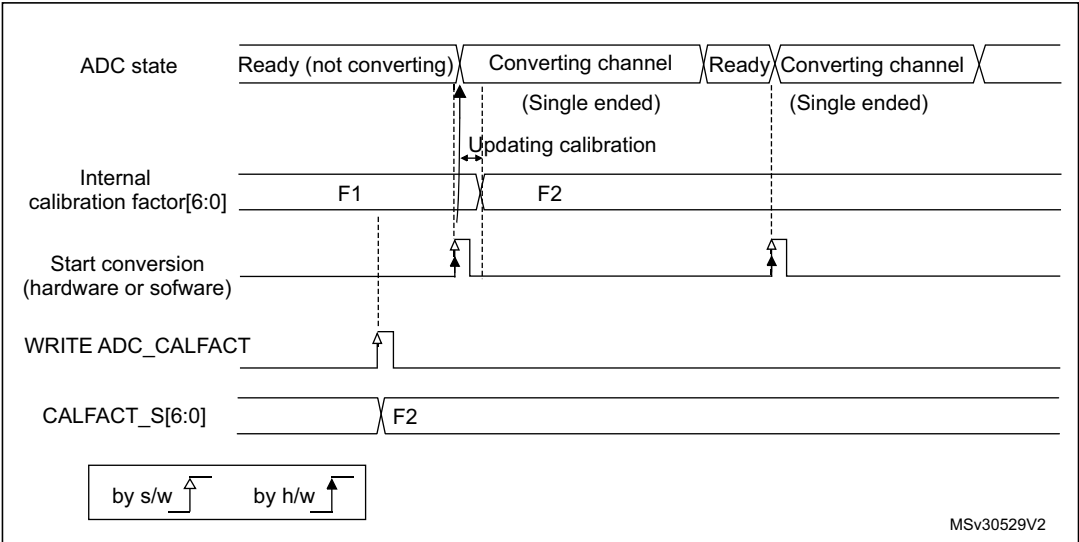
Figure 54. ADC calibration



Software procedure to re-inject a calibration factor into the ADC

1. Ensure ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor will be injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Figure 55. Updating the ADC calibration factor

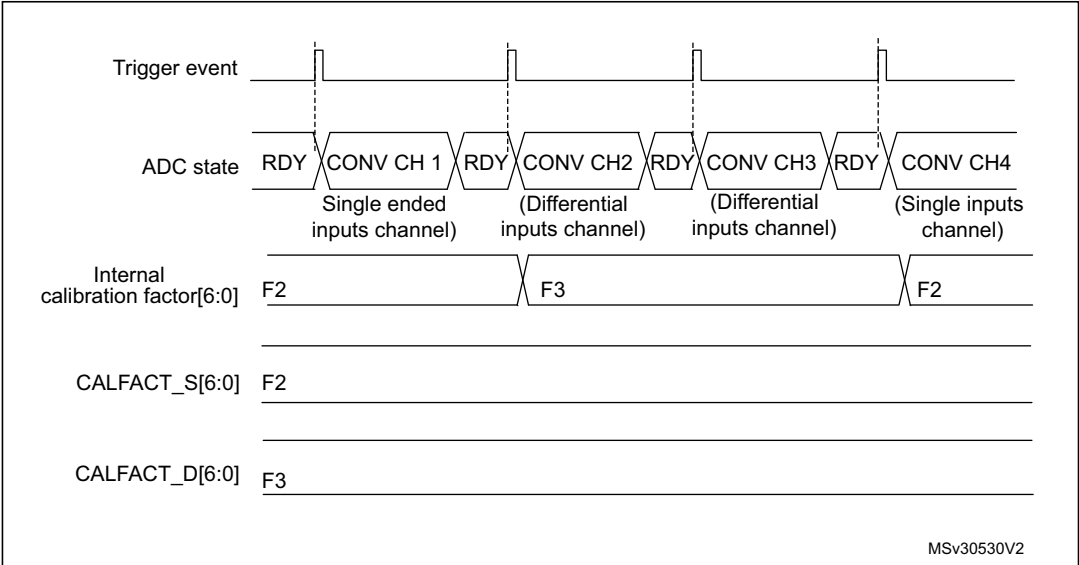


Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF=0 and one with ADCALDIF=1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF=0). This updates the register CALFACT_S[6:0].
3. Calibrate the ADC in Differential input modes (with ADCALDIF=1). This updates the register CALFACT_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration will automatically be injected into the analog ADC.

Figure 56. Mixing single-ended and differential channels



15.3.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in [Section 15.3.6: ADC voltage regulator \(ADVREGEN\)](#).

Once ADVREGEN[1:0] = 01, the ADC can be enabled and the ADC needs a stabilization time of t_{STAB} before it starts converting accurately, as shown in [Figure 57](#). Two control bits enable or disable the ADC:

- ADEN=1 enables the ADC. The flag ADRDY will be set once the ADC is ready for operation.
- ADDIS=1 disables the ADC and disable the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART=1 (refer to [Section 15.3.18: Conversion on external trigger and trigger polarity \(EXTSEL, EXTEN, JEXTSEL, JEXTEN\)](#)) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART=1 or when an external injected trigger event occurs, if injected triggers are enabled.

Software procedure to enable the ADC

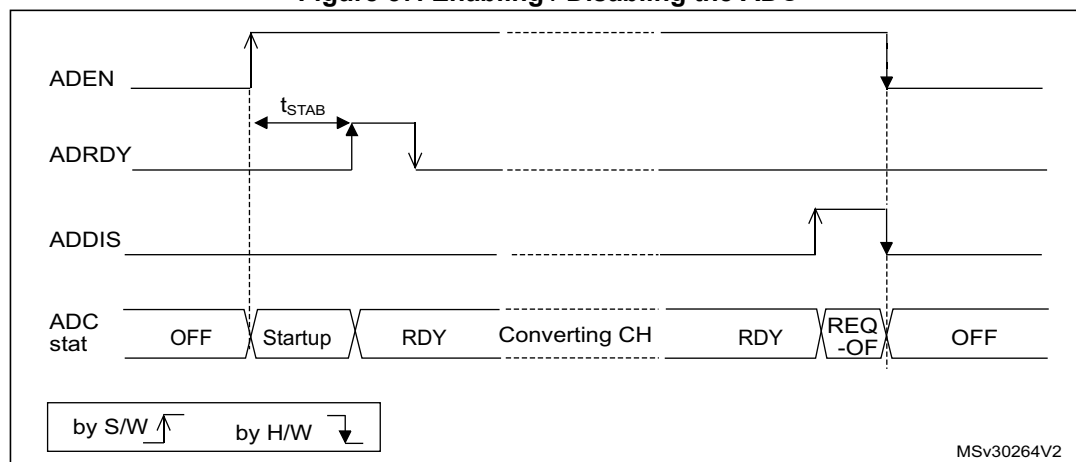
1. Set ADEN=1.
2. Wait until ADRDY=1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE=1).

Note: ADEN bit cannot be set during ADCAL=1 and 4 ADC clock cycle after the ADCAL bit is cleared by hardware(end of the calibration).

Software procedure to disable the ADC

1. Check that both ADSTART=0 and JADSTART=0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP=1 and JADSTP=1 and then wait until ADSTP=0 and JADSTP=0.
2. Set ADDIS=1.
3. If required by the application, wait until ADEN=0, until the analog ADC is effectively disabled (ADDIS will automatically be reset once ADEN=0).

Figure 57. Enabling / Disabling the ADC



15.3.10 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the control bits DIFSEL in the ADCx_DIFSEL register and the control bits ADCAL and ADEN in the ADCx_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADCx_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADCx_CFGR, ADCx_SMPRx, ADCx_TRx, ADCx_SQRx, ADCx_JDRy, ADCx_OFRRy, ADCx_OFCHR and ADCx_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).

The software is allowed to write the control bits ADSTP or JADSTP of the ADCx_CR register only if the ADC is enabled and eventually converting and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADCx_JSQR at any time, when the ADC is enabled (ADEN=1).

Note: There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN=0 as well as all the bits of ADCx_CR register).

15.3.11 Channel selection (SQRx, JSQRx)

There are up to 18 multiplexed channels per ADC:

- Up to 5 fast analog inputs coming from GPIO pads (ADC_IN1..5)
- Up to 10 slow analog inputs coming from GPIO pads (ADC_IN5..15). Depending on the products, not all of them are available on GPIO pads.
- ADC1 is connected to 4 internal analog inputs:
 - ADC1_IN15 = $V_{REFOPAMP1}$ = Reference Voltage for the Operational Amplifier 1
 - ADC1_IN16 = V_{TS} = Temperature Sensor
 - ADC1_IN17 = $V_{BAT}/2 = V_{BAT}$ channel
 - ADC1_IN18 = V_{REFINT} = Internal Reference Voltage (also connected to ADC2_IN18).
- ADC2_IN17 = $V_{REFOPAMP2}$ = Reference Voltage for the Operational Amplifier 2

Warning: The user must ensure that only one of the two ADCs is converting V_{REFINT} at the same time (it is forbidden to have several ADCs converting V_{REFINT} at the same time).

Note: To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, TSEN or VBATEN in the ADCx_CCR registers.

It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADC_IN3, ADC_IN8, ADC_IN2, ADC_IN2, ADC_IN0, ADC_IN2, ADC_IN2, ADC_IN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADCx_SQR registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADCx_SQR1 register.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADCx_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADCx_JSQR register.

ADCx_SQR registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP=1 (refer to [Section 15.3.17: Stopping an ongoing conversion \(ADSTP, JADSTP\)](#)).

It is possible to modify the ADCx_JSQR registers on-the-fly while injected conversions are occurring. Refer to [Section 15.3.21: Queue of context for injected conversions](#)

15.3.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADCx_SMPR1 and ADCx_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 1.5 ADC clock cycles
- SMP = 001: 2.5 ADC clock cycles
- SMP = 010: 4.5 ADC clock cycles
- SMP = 011: 7.5 ADC clock cycles
- SMP = 100: 19.5 ADC clock cycles
- SMP = 101: 61.5 ADC clock cycles
- SMP = 110: 181.5 ADC clock cycles
- SMP = 111: 601.5 ADC clock cycles

The total conversion time is calculated as follows:

$$T_{\text{conv}} = \text{Sampling time} + 12.5 \text{ ADC clock cycles}$$

Example:

With $F_{\text{ADC_CLK}} = 72 \text{ MHz}$ and a sampling time of 1.5 ADC clock cycles:

$$T_{\text{conv}} = (1.5 + 12.5) \text{ ADC clock cycles} = 14 \text{ ADC clock cycles} = 0.194 \mu\text{s (for fast channels)}$$

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Constraints on the sampling time for fast and slow channels

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.

15.3.13 Single conversion mode (CONT=0)

In Single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADCx_CR register (for a regular channel)
- Setting the JADSTART bit in the ADCx_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADCx_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 16-bit ADCx_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

Note: To convert a single channel, program a sequence with a length of 1.

15.3.14 Continuous conversion mode (CONT=1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically re-starts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADCx_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADCx_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to [Auto-injection mode](#) section).

15.3.15 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART=1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN = 0x0 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN != 0x0

Software starts ADC injected conversions by setting JADSTART=1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN = 0x0 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN != 0x0

Note: In auto-injection mode (JAUTO=1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART=0 and JADSTART=0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software regular trigger (CONT=0, EXTSEL=0x0)
 - at any end of regular conversion sequence (EOS assertion) or at any end of sub-group processing if DISCEN = 1
- In all cases (CONT=x, EXTSEL=x)
 - after execution of the ADSTP procedure asserted by the software.

Note: In continuous mode (CONT=1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT=0 and EXTSEL != 0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is cleared by hardware:

- in single mode with software injected trigger (JEXTSEL=0x0)
 - at any end of injected conversion sequence (JEOS assertion) or at any end of sub-group processing if JDISCEN = 1
- in all cases (JEXTSEL=x)
 - after execution of the JADSTP procedure asserted by the software.

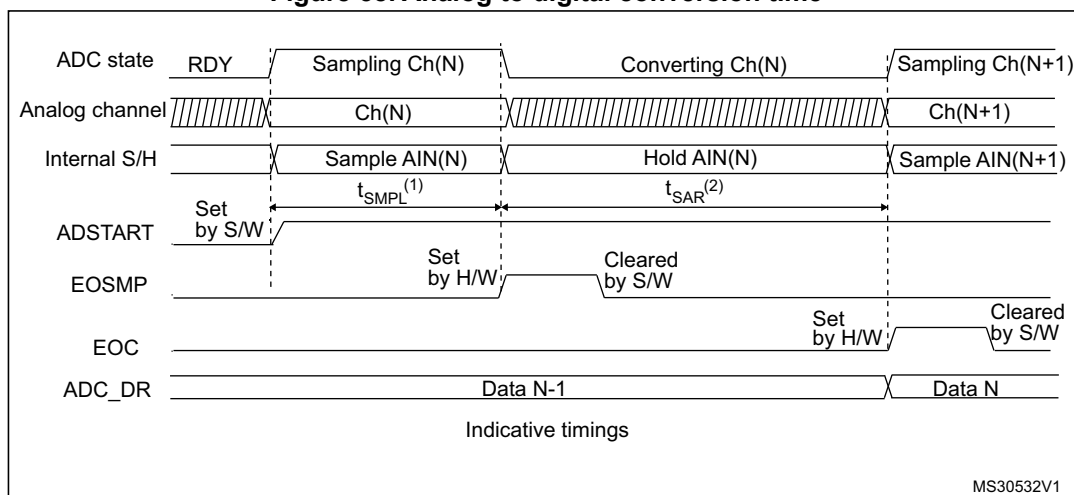
15.3.16 Timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$$T_{ADC} = T_{SMPL} + T_{SAR} = [1.5_{|min} + 12.5_{|12bit}] \times T_{ADC_CLK}$$

$$T_{ADC} = T_{SMPL} + T_{SAR} = 20.83 \text{ ns}_{|min} + 173.6 \text{ ns}_{|12bit} = 194.4 \text{ ns (for } F_{ADC_CLK} = 72 \text{ MHz)}$$

Figure 58. Analog to digital conversion time



MS30532V1

1. T_{SMPL} depends on SMP[2:0]

2. T_{SAR} depends on RES[2:0]

15.3.17 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP=1 and injected conversions ongoing by setting JADSTP=1.

Stopping conversions will reset the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADCx_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADCx_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would re-start a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must wait until ADSTART = 0 (or JADSTART = 0) before starting a new conversion.

Note: *In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).*

Figure 59. Stopping ongoing regular conversions

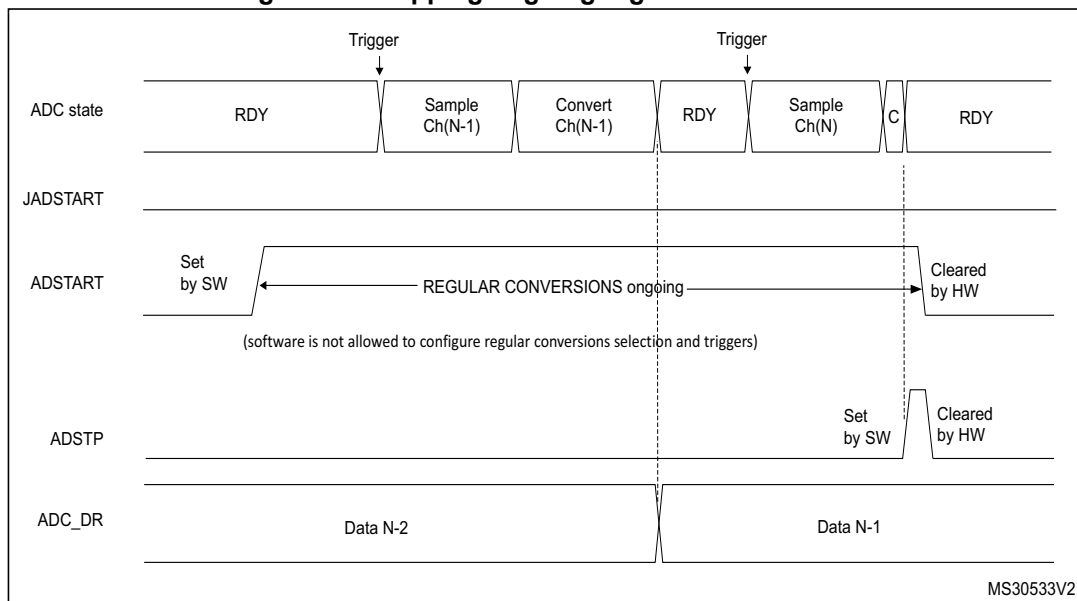
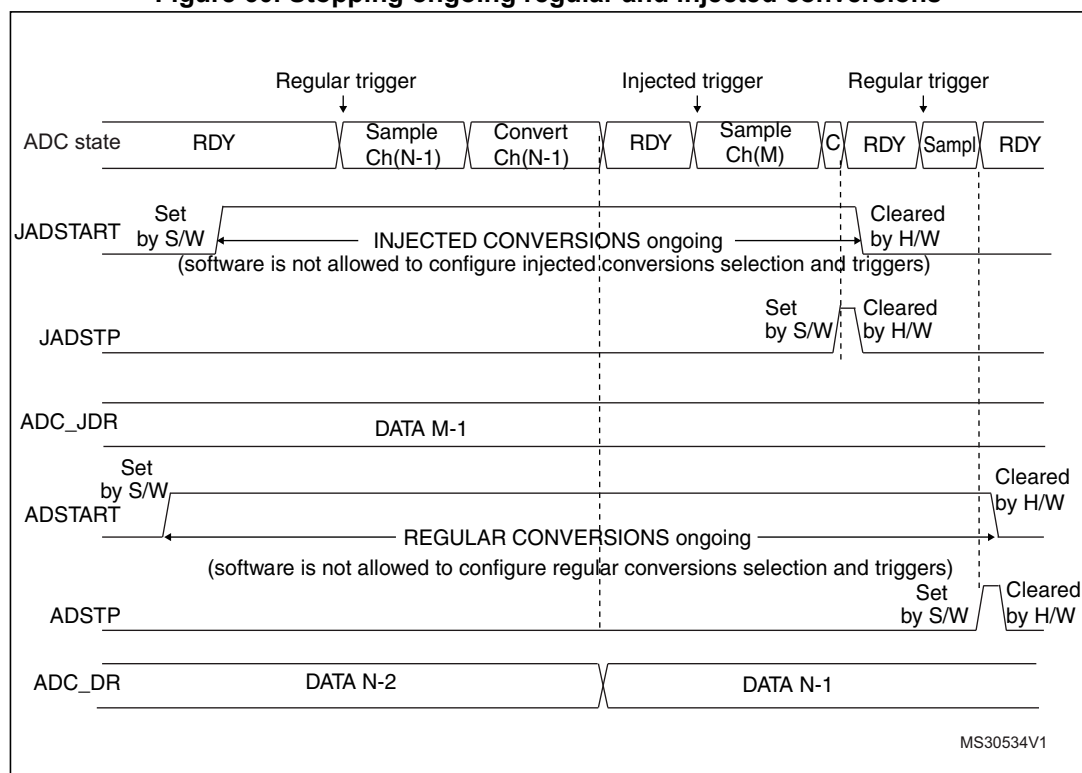


Figure 60. Stopping ongoing regular and injected conversions



15.3.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (e.g. timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

The regular trigger selection is effective once software has set bit ADSTART=1 and the injected trigger selection is effective once software has set bit JADSTART=1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART=0, any regular hardware triggers which occur are ignored.
- If bit JADSTART=0, any injected hardware triggers which occur are ignored.

Table 84 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 84. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

Table 85. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	Hardware Trigger with detection on the rising edge
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the injected trigger can be anticipated and changed on-the-fly. Refer to [Section 15.3.21: Queue of context for injected conversions](#).

The EXTSEL[3:0] and JEXTSEL[3:0] control bits select which out of 16 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Note: The regular trigger selection cannot be changed on-the-fly.
The injected trigger selection can be anticipated and changed on-the-fly. Refer to [Section 15.3.21: Queue of context for injected conversions on page 310](#)

Each ADC master shares the same input triggers with its ADC slave as described in [Figure 61](#).

Figure 61. Triggers are shared between ADC master & ADC slave

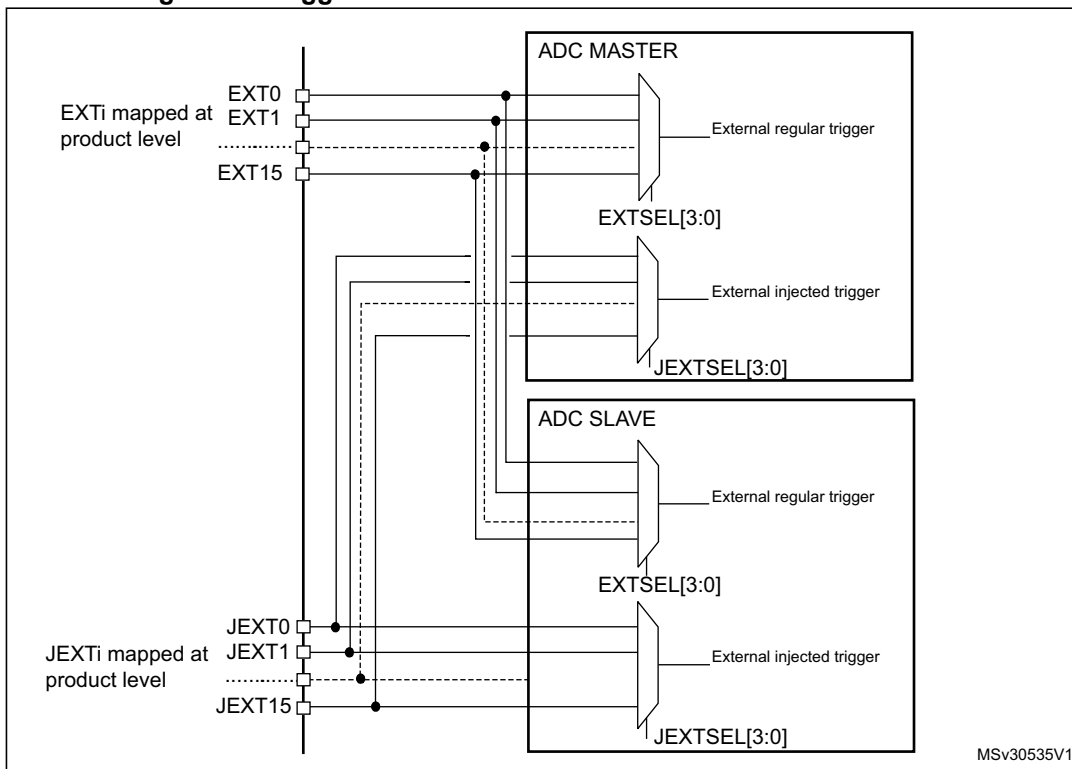


Table 86 to Table 87 give all the possible external triggers of the two ADCs for regular and injected conversion.

Table 86. ADC1 (master) & 2 (slave) - External triggers for regular channels

Name	Source	Type	EXTSEL[3:0]
EXT0	TIM1_CC1 event	Internal signal from on chip timers	0000
EXT1	TIM1_CC2 event	Internal signal from on chip timers	0001
EXT2	TIM1_CC3 event	Internal signal from on chip timers	0010
EXT3	TIM2_CC2 event	Internal signal from on chip timers	0011
EXT4	TIM3_TRGO event	Internal signal from on chip timers	0100
EXT5	TIM4_CC4 event	Internal signal from on chip timers	0101
EXT6	EXTI line 11	External pin	0110
EXT7	Reserved		0111
EXT8	Reserved		1000
EXT9	TIM1_TRGO event	Internal signal from on chip timers	1001
EXT10	TIM1_TRGO2 event	Internal signal from on chip timers	1010
EXT11	TIM2_TRGO event	Internal signal from on chip timers	1011
EXT12	TIM4_TRGO event	Internal signal from on chip timers	1100
EXT13	TIM6_TRGO event	Internal signal from on chip timers	1101
EXT14	TIM15_TRGO event	Internal signal from on chip timers	1110
EXT15	TIM3_CC4 event	Internal signal from on chip timers	1111

Table 87. ADC1 & ADC2 - External trigger for injected channels

Name	Source	Type	JEXTSEL[3..0]
JEXT0	TIM1_TRGO event	Internal signal from on chip timers	0000
JEXT1	TIM1_CC4 event	Internal signal from on chip timers	0001
JEXT2	TIM2_TRGO event	Internal signal from on chip timers	0010
JEXT3	TIM2_CC1 event	Internal signal from on chip timers	0011
JEXT4	TIM3_CC4 event	Internal signal from on chip timers	0100
JEXT5	TIM4_TRGO event	Internal signal from on chip timers	0101
JEXT6	EXTI line 15	External pin	0110
JEXT7	Reserved	-	0111
JEXT8	TIM1_TRGO2 event	Internal signal from on chip timers	1000
JEXT9	Reserved		1001
JEXT10	Reserved		1010
JEXT11	TIM3_CC3 event	Internal signal from on chip timers	1011
JEXT12	TIM3_TRGO event	Internal signal from on chip timers	1100

Table 87. ADC1 & ADC2 - External trigger for injected channels

Name	Source	Type	JEXTSEL[3..0]
JEXT13	TIM3_CC1 event	Internal signal from on chip timers	1101
JEXT14	TIM6_TRGO event	Internal signal from on chip timers	1110
JEXT15	TIM15_TRGO event	Internal signal from on chip timers	1111

15.3.19 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADCx_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADCx_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADCx_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence.

Figure 62 shows the corresponding timing diagram.

Note: *When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 28 ADC clock cycles (that is two conversions with a sampling time of 1.5 clock periods), the minimum interval between triggers must be 29 ADC clock cycles.*

Auto-injection mode

If the JAUTO bit in the ADCx_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADCx_SQR and ADCx_JSQR registers.

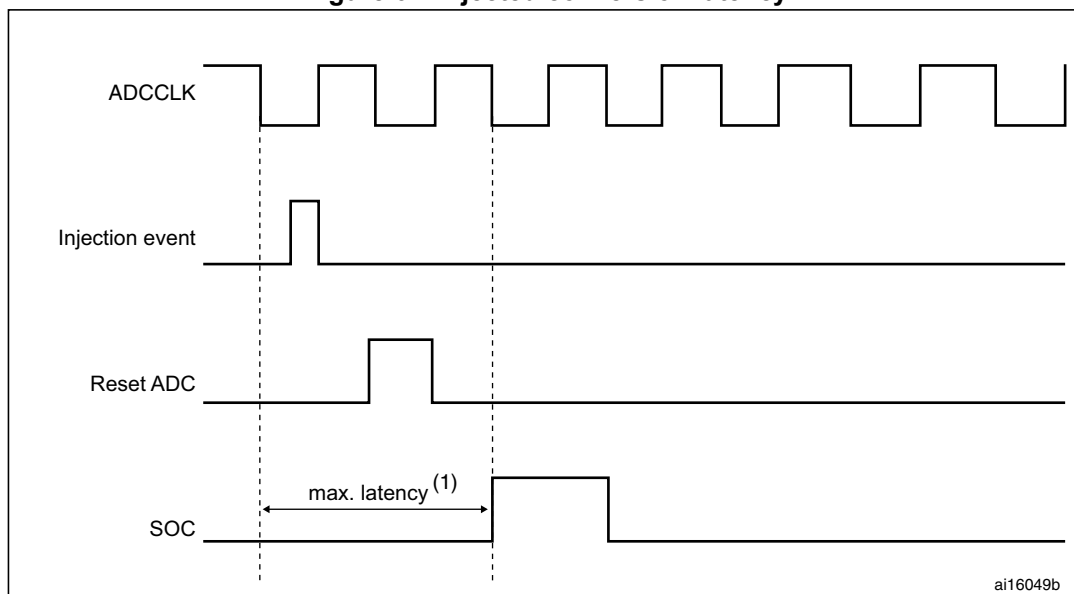
In this mode, the ADSTART bit in the ADCx_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: *It is not possible to use both the auto-injected and discontinuous modes simultaneously. When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA_CCRx register). If the CIRC bit is reset (single-shot mode), the JAUTO sequence will be stopped upon DMA Transfer Complete event.*

Figure 62. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the STM32F302xx datasheets.

15.3.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADCx_CFGR register.

It is used to convert a short sequence (sub-group) of n conversions ($n \leq 8$) that is part of the sequence of conversions selected in the ADCx_SQR registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADCx_CFGR register.

When an external trigger occurs, it starts the next n conversions selected in the ADCx_SQR registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADCx_SQR1 register.

Example:

- DISCEN=1, $n=3$, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
 - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
 - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
 - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
 - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
 - ...
- DISCEN=0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
 - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
 - all the next trigger events will relaunch the complete sequence.

Note: *When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).*

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN=1, CONT=1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADCx_CFGR register. It converts the sequence selected in the ADCx_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where 'n' is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADCx_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADCx_JSQR register.

Example:

- JDISCEN=1, channels to be converted = 1, 2, 3
 - 1st trigger: channel 1 converted (a JEOC event is generated)
 - 2nd trigger: channel 2 converted (a JEOC event is generated)
 - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
 - ...

Note: *When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.*

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

15.3.21 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL[3:0] in ADCx_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADCx_JSQR register)

All the parameters of the context are defined into a single register ADCx_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADCx_CFGR:
 - If JQM=0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence will be served according to the last active context.
 - If JQM=1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and both injected software and hardware triggers are disabled. Therefore, any further hardware or software injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP=1 or when disabling the ADC by setting ADDIS=1:
 - If JQM=0, the Queue is maintained with the last active context.
 - If JQM=1, the Queue becomes empty and triggers are ignored.

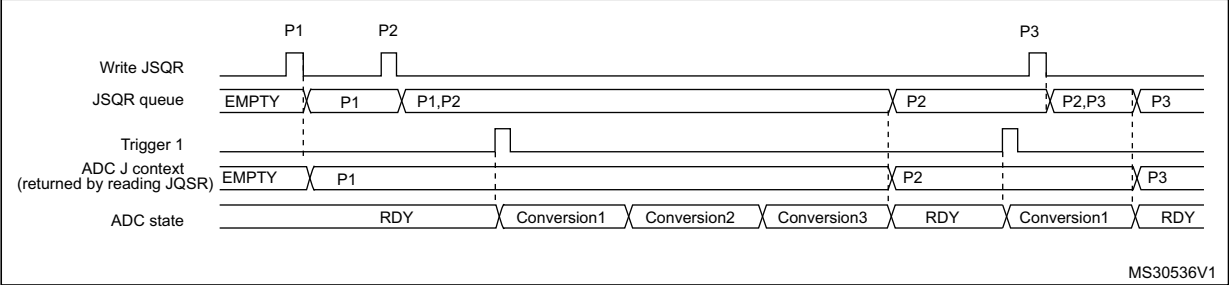
Note: *When configured in discontinuous mode (bit JDISCEN=1), only the last trigger of the injected sequence changes the context and consumes the Queue. The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):*

- 1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out
- 2nd trigger, disc. Sequence 1: 2nd conversion.
- 3rd trigger, discontinuous. Sequence 1: 3rd conversion.
- 4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.
- 5th trigger, discontinuous. Sequence 2: 2nd conversion.
- 6th trigger, discontinuous. Sequence 2: 3rd conversion.

Behavior when changing the trigger or sequence context

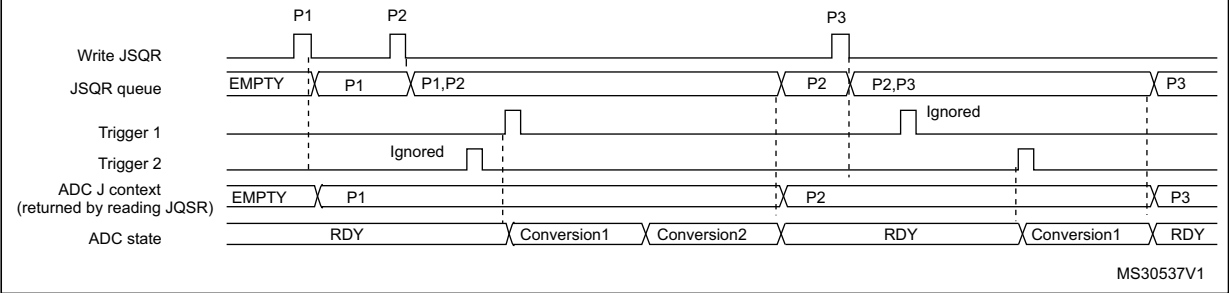
The [Figure 63](#) and [Figure 64](#) show the behavior of the context Queue when changing the sequence or the triggers.

Figure 63. Example of JSQR queue of context (sequence change)



1. Parameters:
- P1: sequence of 3 conversions, hardware trigger 1
 - P2: sequence of 1 conversion, hardware trigger 1
 - P3: sequence of 4 conversions, hardware trigger 1

Figure 64. Example of JSQR queue of context (trigger change)

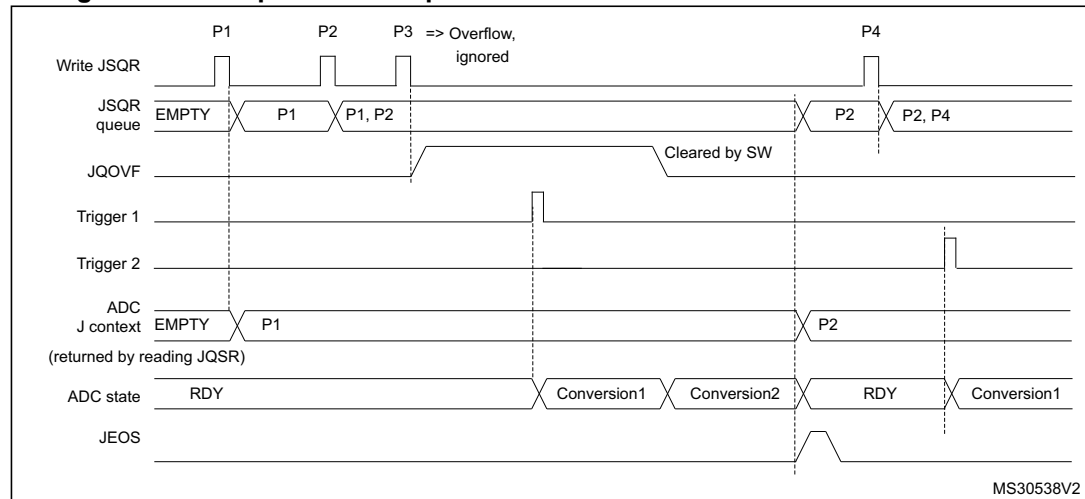


1. Parameters:
- P1: sequence of 2 conversions, hardware trigger 1
 - P2: sequence of 1 conversion, hardware trigger 2
 - P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

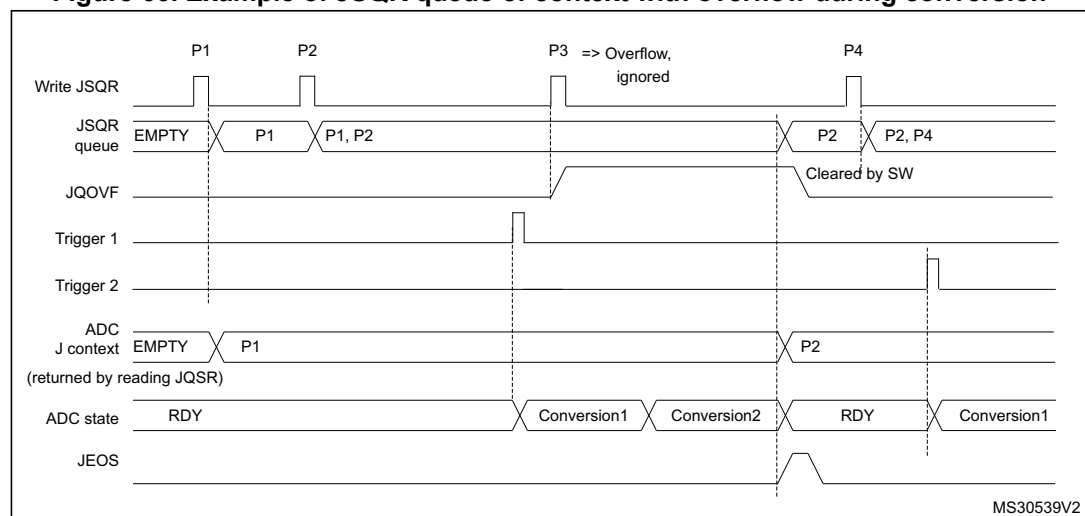
The [Figure 65](#) and [Figure 66](#) show the behavior of the context Queue if an overflow occurs before or during a conversion.

Figure 65. Example of JSQR queue of context with overflow before conversion



- Parameters:
 - P1: sequence of 2 conversions, hardware trigger 1
 - P2: sequence of 1 conversion, hardware trigger 2
 - P3: sequence of 3 conversions, hardware trigger 1
 - P4: sequence of 4 conversions, hardware trigger 1

Figure 66. Example of JSQR queue of context with overflow during conversion



- Parameters:
 - P1: sequence of 2 conversions, hardware trigger 1
 - P2: sequence of 1 conversion, hardware trigger 2
 - P3: sequence of 3 conversions, hardware trigger 1
 - P4: sequence of 4 conversions, hardware trigger 1

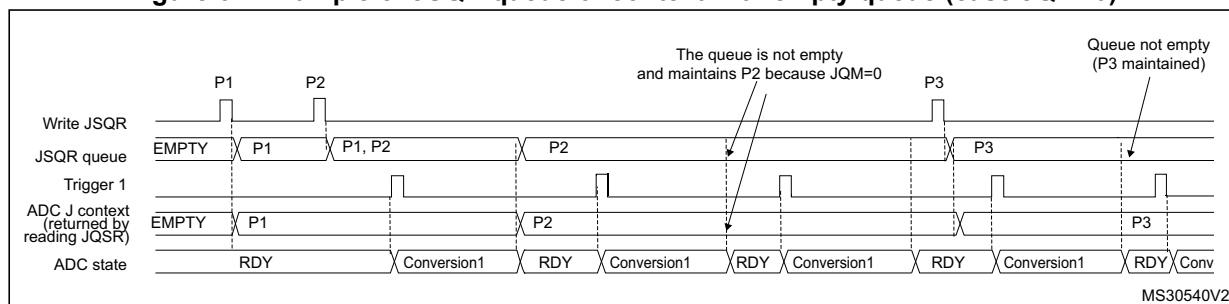
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

Figure 67 and Figure 68 show the behavior of the context Queue when the Queue becomes empty in both cases JQM=0 or 1.

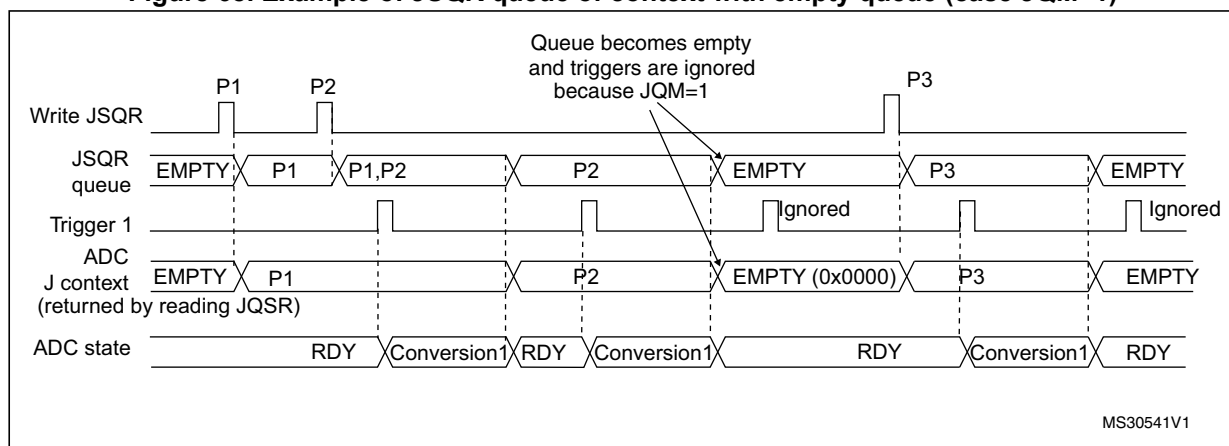
Figure 67. Example of JSQR queue of context with empty queue (case JQM=0)



- Parameters:
 - P1: sequence of 1 conversion, hardware trigger 1
 - P2: sequence of 1 conversion, hardware trigger 1
 - P3: sequence of 1 conversion, hardware trigger 1

Note: When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.

Figure 68. Example of JSQR queue of context with empty queue (case JQM=1)

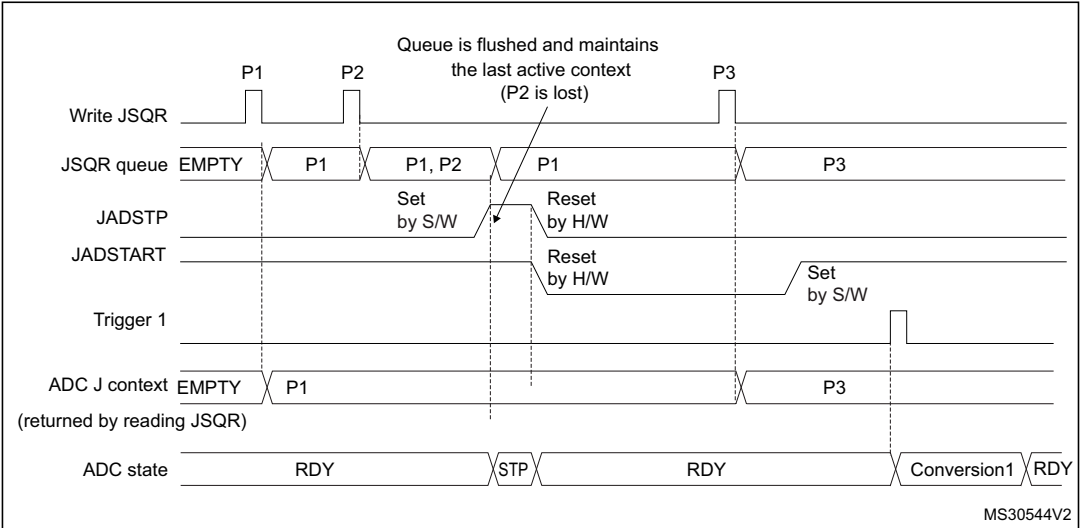


- Parameters:
 - P1: sequence of 1 conversion, hardware trigger 1
 - P2: sequence of 1 conversion, hardware trigger 1
 - P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

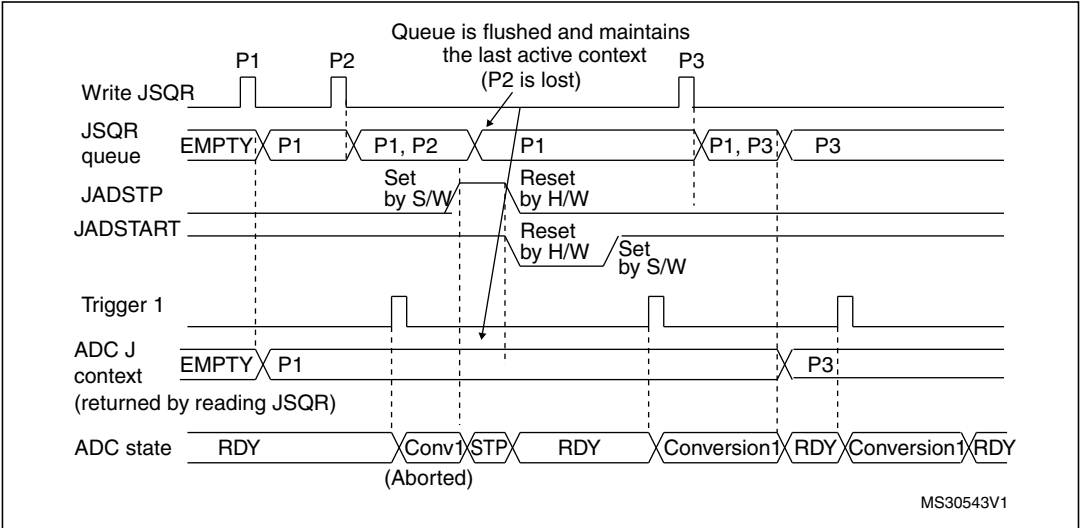
The figures below show the behavior of the context Queue in various situations when the queue is flushed.

**Figure 69. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0).
Case when JADSTP occurs during an ongoing conversion.**



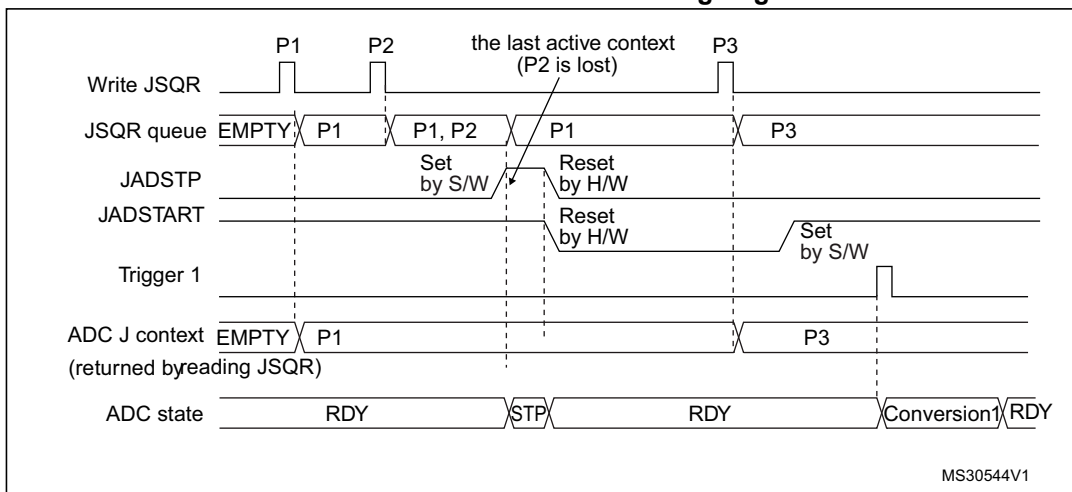
- Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

**Figure 70. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0).
Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.**



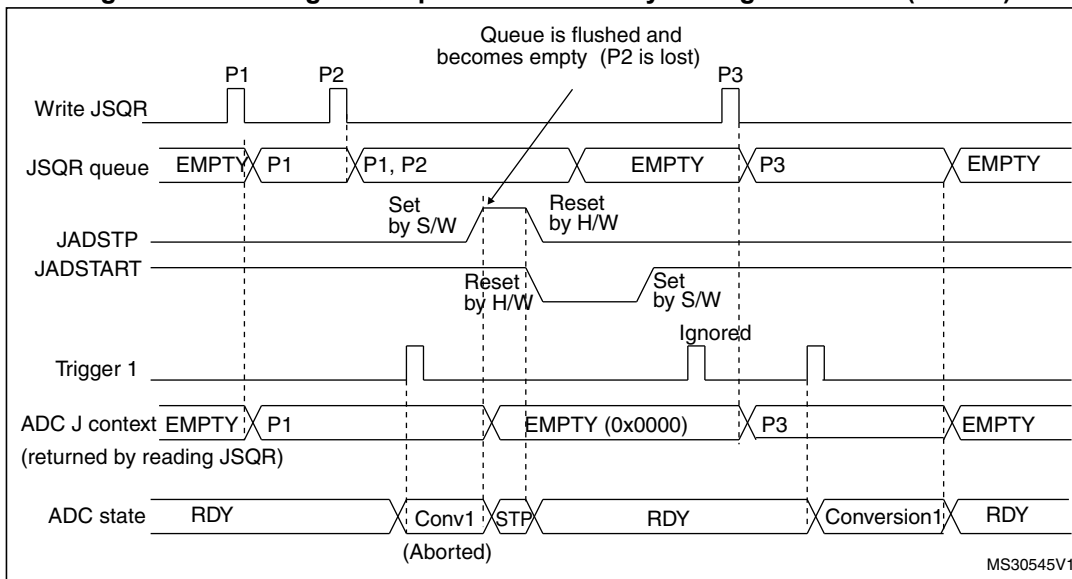
- Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

**Figure 71. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0).
Case when JADSTP occurs outside an ongoing conversion**

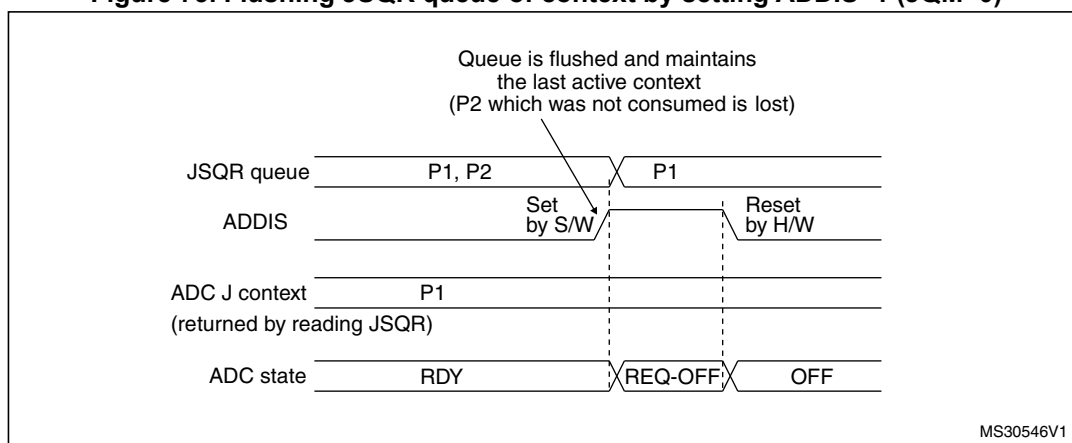


- Parameters:
 - P1: sequence of 1 conversion, hardware trigger 1
 - P2: sequence of 1 conversion, hardware trigger 1
 - P3: sequence of 1 conversion, hardware trigger 1

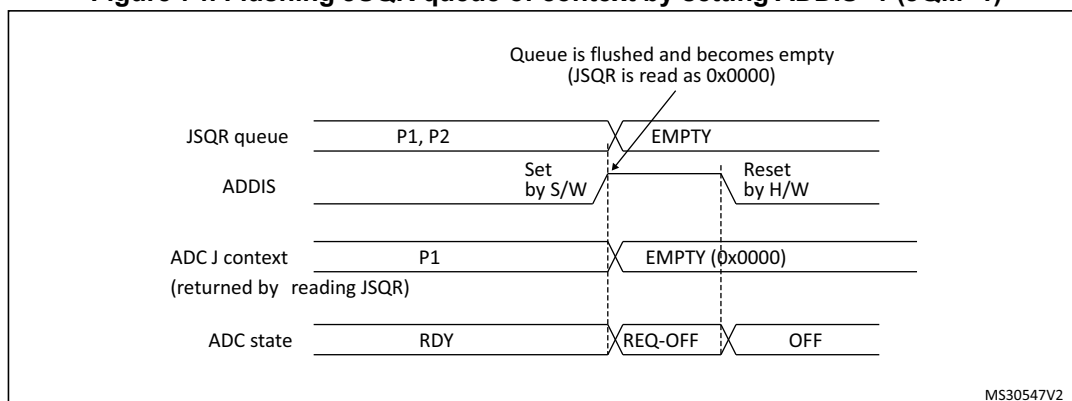
Figure 72. Flushing JSQR queue of context by setting JADSTP=1 (JQM=1)



- Parameters:
 - P1: sequence of 1 conversion, hardware trigger 1
 - P2: sequence of 1 conversion, hardware trigger 1
 - P3: sequence of 1 conversion, hardware trigger 1

Figure 73. Flushing JSQR queue of context by setting ADDIS=1 (JQM=0)

- Parameters:
 P1: sequence of 1 conversion, hardware trigger 1
 P2: sequence of 1 conversion, hardware trigger 1
 P3: sequence of 1 conversion, hardware trigger 1

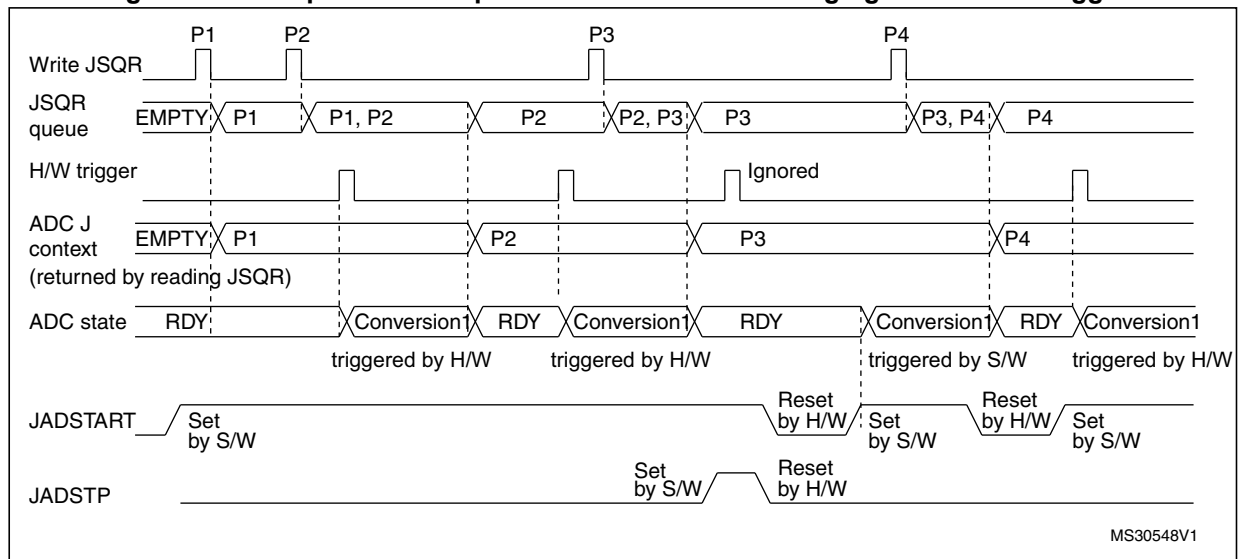
Figure 74. Flushing JSQR queue of context by setting ADDIS=1 (JQM=1)

- Parameters:
 P1: sequence of 1 conversion, hardware trigger 1
 P2: sequence of 1 conversion, hardware trigger 1
 P3: sequence of 1 conversion, hardware trigger 1

Changing context from hardware to software (or software to hardware) injected trigger

When changing the context from hardware trigger to software injected trigger, it is necessary to stop the injected conversions by setting JADSTP=1 after the last hardware triggered conversions. This is necessary to re-enable the software trigger (a rising edge on JADSTART is necessary to start a software injected conversion). Refer to [Figure 75](#).

When changing the context from software trigger to hardware injected trigger, after the last software trigger, it is necessary to set JADSTART=1 to enable the hardware triggers. Refer to [Figure 75](#).

Figure 75. Example of JSQR queue of context when changing SW and HW triggers

1. Parameters:
 - P1: sequence of 1 conversion, hardware trigger (JEXTEN != 0x0)
 - P2: sequence of 1 conversion, hardware trigger (JEXTEN != 0x0)
 - P3: sequence of 1 conversion, software trigger (JEXTEN = 0x0)
 - P4: sequence of 1 conversion, hardware trigger (JEXTEN != 0x0)

Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy JSQR with JEXTEN not equal to 0 (otherwise triggering a software conversion)
6. Set JADSTART
7. Set JADSTP
8. Wait until JADSTART is reset
9. Set JADSTART.

15.3.22 Programmable resolution (RES) - fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the control bits RES[1:0]. [Figure 80](#), [Figure 81](#), [Figure 82](#) and [Figure 83](#) show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to [Table 88](#).

Table 88. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{ADC}=72$ MHz	T_{ADC} (ADC clock cycles) (with Sampling Time= 1.5 ADC clock cycles)	T_{ADC} (ns) at $F_{ADC}=72$ MHz
12	12.5 ADC clock cycles	173.6 ns	14 ADC clock cycles	194.4 ns
10	10.5 ADC clock cycles	145.8 ns	12 ADC clock cycles	166.7 ns
8	8.5 ADC clock cycles	118.0 ns	10 ADC clock cycles	138.9 ns
6	6.5 ADC clock cycles	90.3 ns	8 ADC clock cycles	111.1 ns

15.3.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADCx_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADCx_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADCx_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADCx_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

15.3.24 End of conversion sequence (EOS, JEOS)

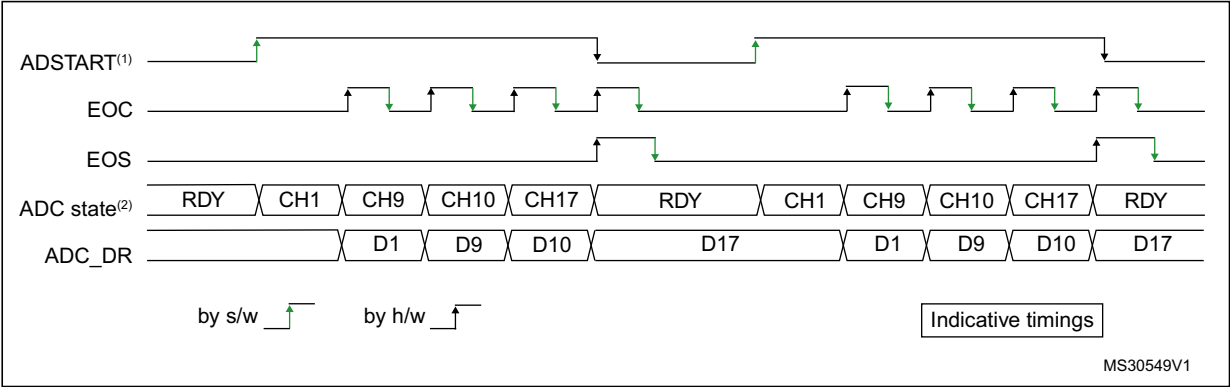
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADCx_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

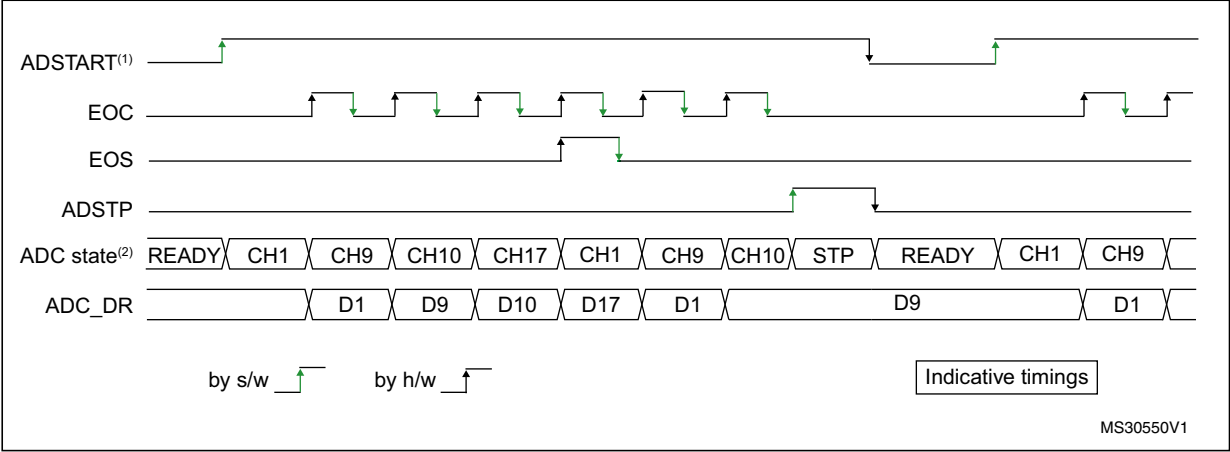
15.3.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

Figure 76. Single conversions of a sequence, software trigger

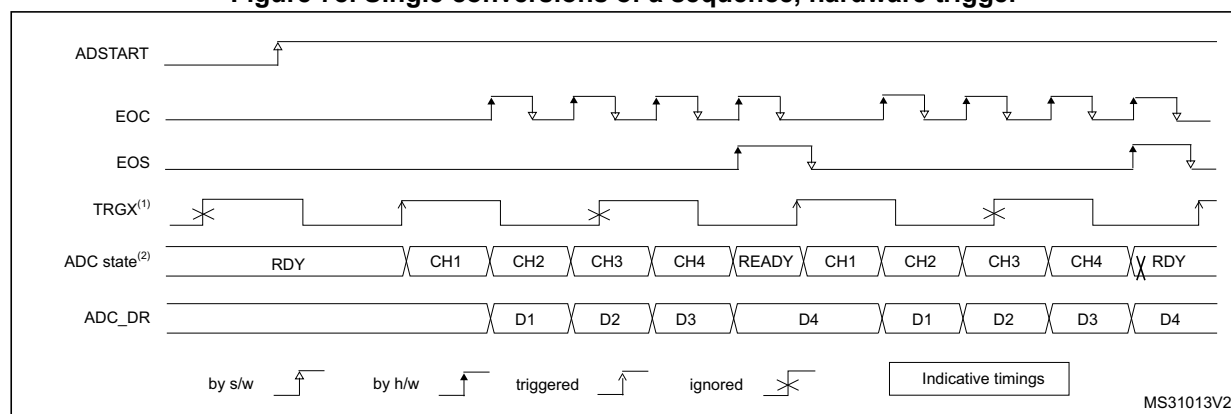


1. EXTEN=0x0, CONT=0
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

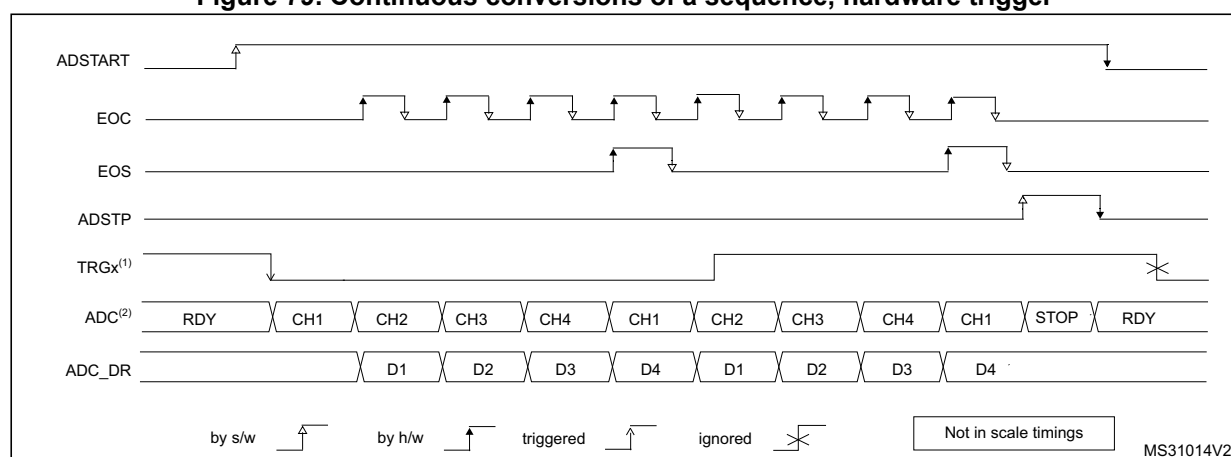
Figure 77. Continuous conversion of a sequence, software trigger



1. EXTEN=0x0, CONT=1
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 78. Single conversions of a sequence, hardware trigger

1. TRGX (over-frequency) is selected as trigger source, EXTEN = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

Figure 79. Continuous conversions of a sequence, hardware trigger

1. TRGX is selected as trigger source, EXTEN = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

15.3.26 Data management

Data register, data alignment and offset (ADCx_DR, OFFSETy, OFFSETy_CH, ALIGN)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADCx_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADCx_JDRy data register which is 16 bits wide.

The ALIGN bit in the ADCx_CFGR register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in [Figure 80](#), [Figure 81](#), [Figure 82](#) and [Figure 83](#).

Special case: when left-aligned, the data are aligned on a half-word basis except when the resolution is set to 6-bit. In that case, the data are aligned on a byte basis as shown in [Figure 82](#) and [Figure 83](#).

Offset

An offset y ($y=1,2,3,4$) can be applied to a channel by setting the bit `OFFSETy_EN=1` into `ADCx_OFRy` register. The channel to which the offset will be applied is programmed into the bits `OFFSETy_CH[4:0]` of `ADCx_OFRy` register. In this case, the converted value is decreased by the user-defined offset written in the bits `OFFSETy[11:0]`. The result may be a negative value so the read data is signed and the `SEXT` bit represents the extended sign value.

[Table 91](#) describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 89. Offset computation versus data resolution

Resolution (bits RES[1:0])	Substraction between raw converted data and offset:		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	signed 12-bit data	-
01: 10-bit	DATA[11:2],00	OFFSET[11:0]	signed 10-bit data	The user must configure OFFSET[1:0] to "00"
10: 8-bit	DATA[11:4],00 00	OFFSET[11:0]	signed 8-bit data	The user must configure OFFSET[3:0] to "0000"
11: 6-bit	DATA[11:6],00 0000	OFFSET[11:0]	signed 6-bit data	The user must configure OFFSET[5:0] to "000000"

When reading data from `ADCx_DR` (regular channel) or from `ADCx_JDRy` (injected channel, $y=1,2,3,4$) corresponding to the channel "i":

- If one of the offsets is enabled (bit `OFFSETy_EN=1`) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

[Figure 80](#), [Figure 81](#), [Figure 82](#) and [Figure 83](#) show alignments for signed and unsigned data.

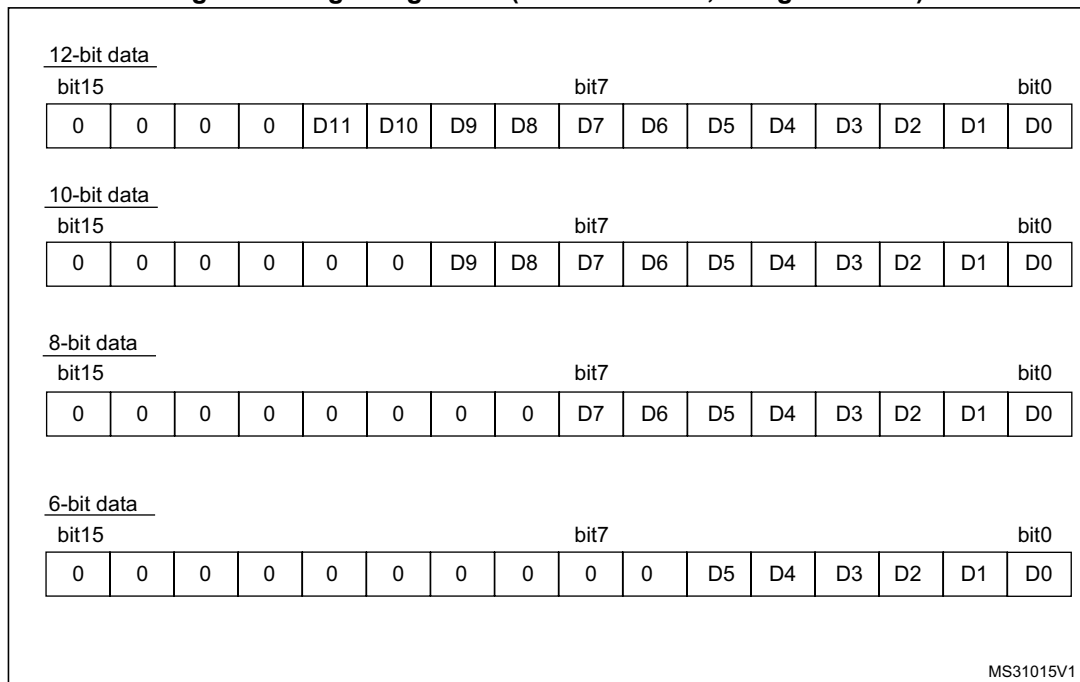
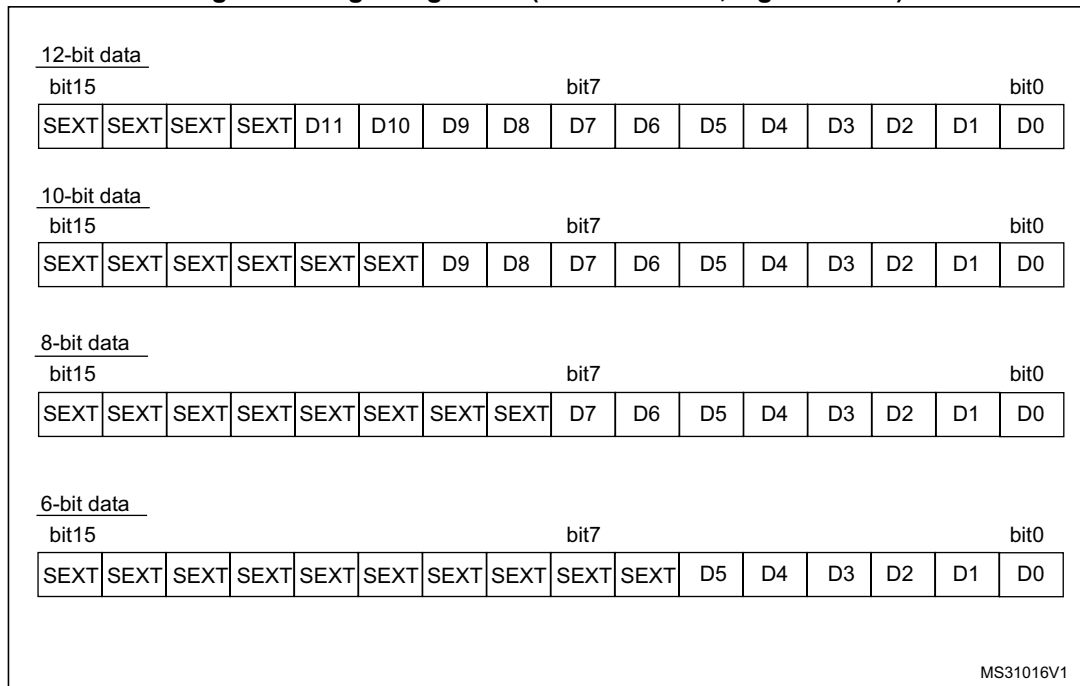
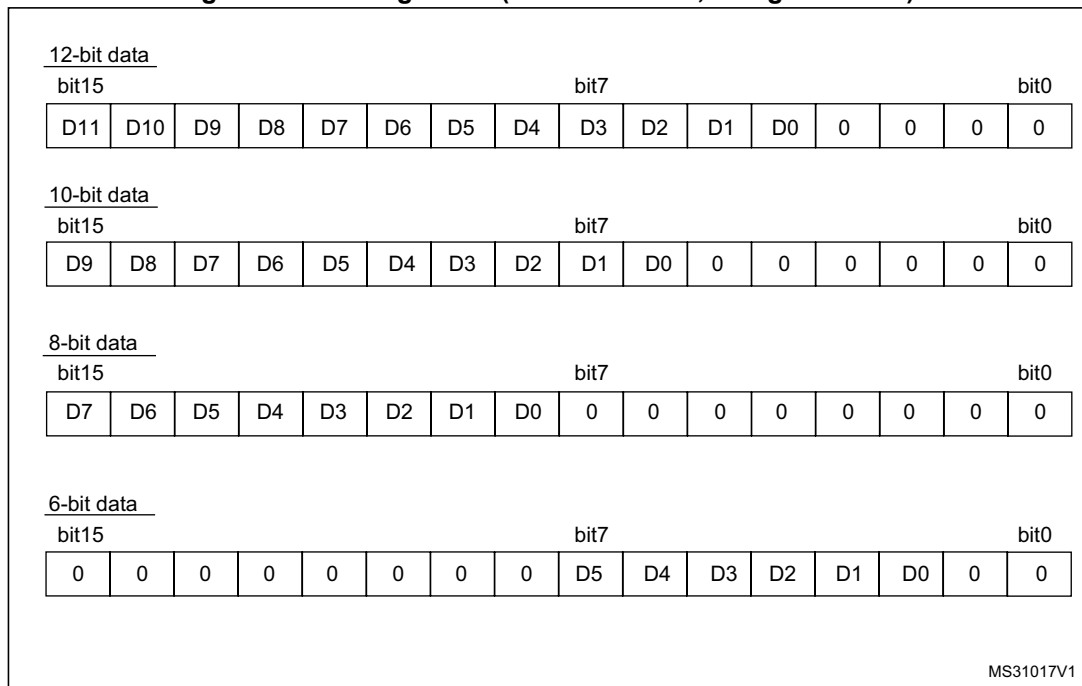
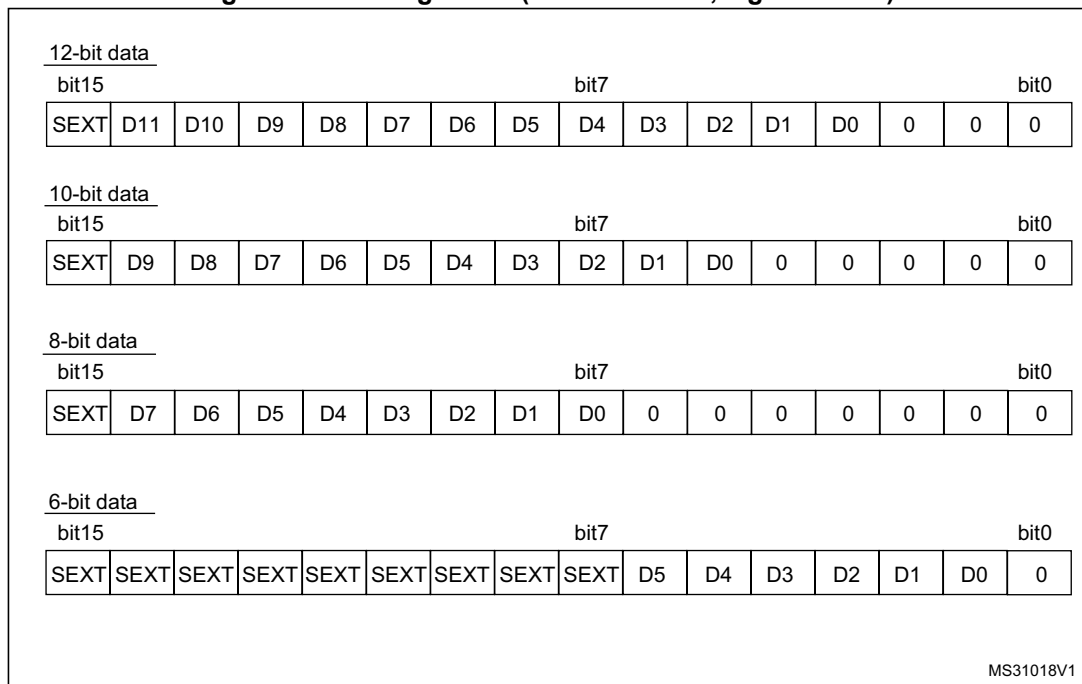
Figure 80. Right alignment (offset disabled, unsigned value)**Figure 81. Right alignment (offset enabled, signed value)**

Figure 82. Left alignment (offset disabled, unsigned value)**Figure 83. Left alignment (offset enabled, signed value)**

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) notifies of a buffer overrun event, when the regular converted data was not read (by the CPU or the DMA) before new converted data became available.

The OVR flag is set if the EOC flag is still 1 at the time when a new conversion completes. An interrupt can be generated if bit OVRIE=1.

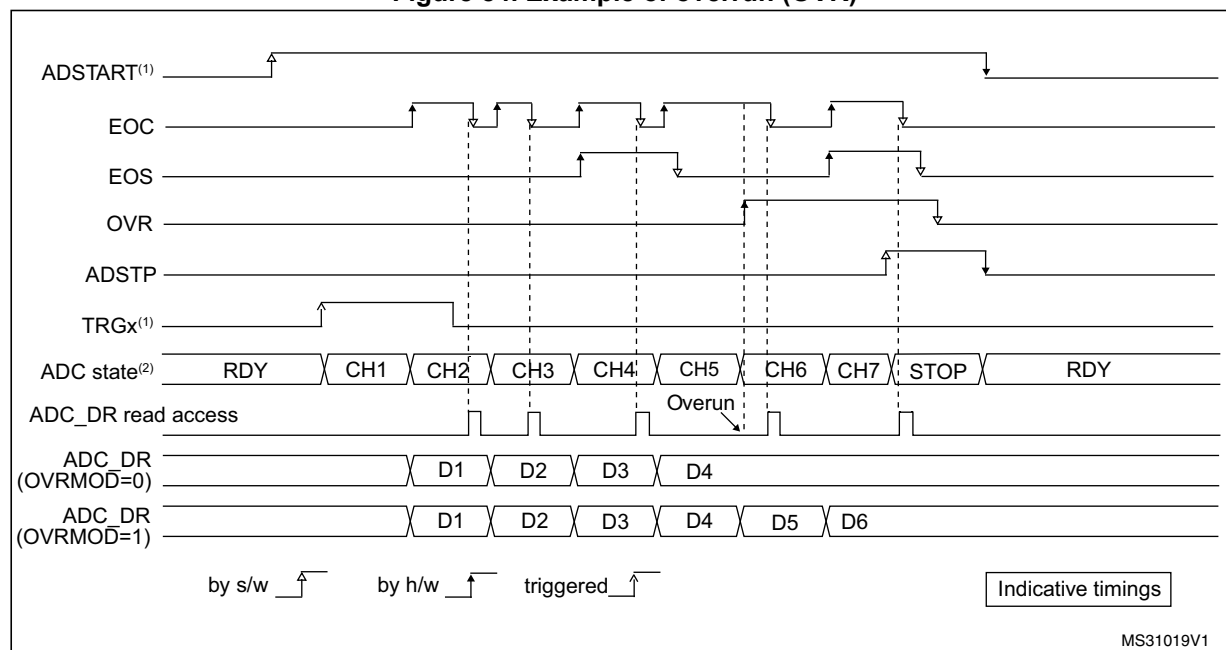
When an overrun condition occurs, the ADC is still operating and can continue to convert unless the software decides to stop and reset the sequence by setting bit ADSTP=1.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD=0: The overrun event preserves the data register from being overrun: the old data is maintained and the new conversion is discarded and lost. If OVR remains at 1, any further conversions will occur but the result data will be also discarded.
- OVRMOD=1: The data register is overwritten with the last conversion result and the previous unread data is lost. If OVR remains at 1, any further conversions will operate normally and the ADCx_DR register will always contain the latest converted data.

Figure 84. Example of overrun (OVR)



Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversion without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADCx_DR register can be read. OVRMOD should be configured to 0 to manage overrun events as an error.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event will not prevent the ADC from continuing to convert and the ADCx_DR register will always contain the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADCx_DR register.

When the DMA mode is enabled (DMAEN bit set to 1 in the ADCx_CFGR register in single ADC mode or MDMA different from 0b00 in dual ADC mode), a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADCx_DR register to the destination location selected by the software.

Despite this, if an overrun occurs (OVR=1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to [Section : ADC overrun \(OVR, OVRMOD\)](#)).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG of the ADCx_CFGR register in single ADC mode, or with bit DMACFG of the ADCx_CCR register in dual ADC mode:

- DMA one shot mode (DMACFG=0).
This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMACFG=1)
This mode is suitable when programming the DMA in circular mode.

DMA one shot mode (DMACFG=0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when DMA_EOT interrupt occurs - refer to DMA paragraph) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.

DMA circular mode (DMACFG=1)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

15.3.27 Dynamic low-power features

Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY=1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADCx_DR register has been read or if the EOC bit has been cleared (see [Figure 85](#)).
- For an injected conversion: when the JEOS bit has been cleared (see [Figure 86](#)).

This is a way to automatically adapt the speed of the ADC to the speed of the system which will read the data.

The delay is inserted after each regular conversion (whatever DISCEN=0 or 1) and after each sequence of injected conversions (whatever JDISCEN=0 or 1).

Note: *There is no delay inserted between each conversions of the injected sequence, except after the last one.*

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

Note: *This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to re-start a conversion: it is up to the software to read the data before launching a new conversion.*

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see [Figure 86](#)).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see [Figure 88](#)).

The behavior is slightly different in auto-injected mode (JAUTO=1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see [Figure 89](#)).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO=1, CONT=1 and AUTDLY=1), follow the following procedure:

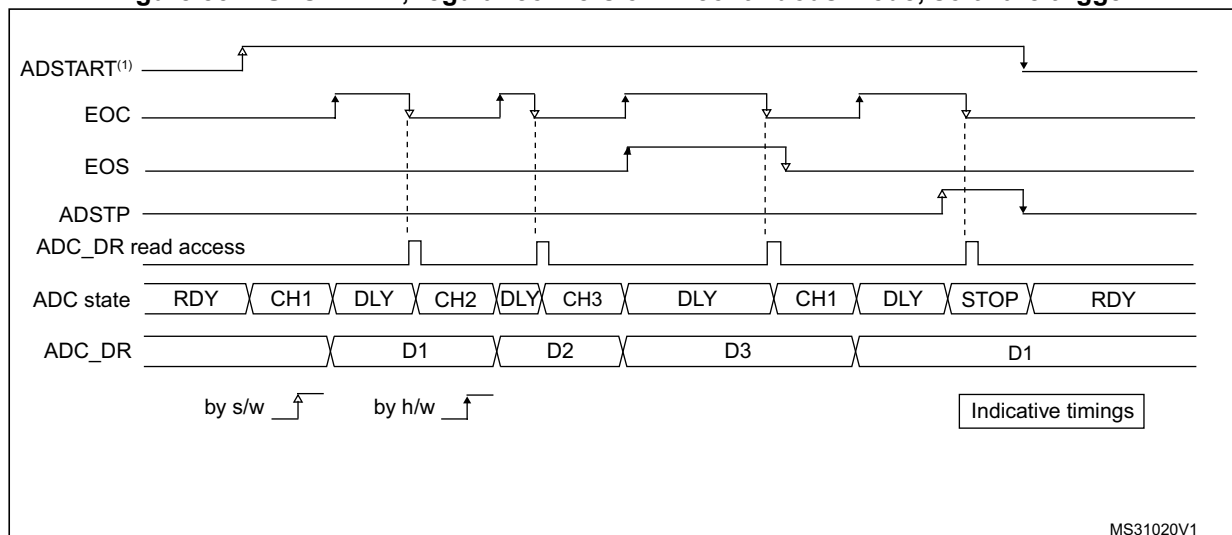
1. Wait until JEOS=1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP=1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can re-start if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence or the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

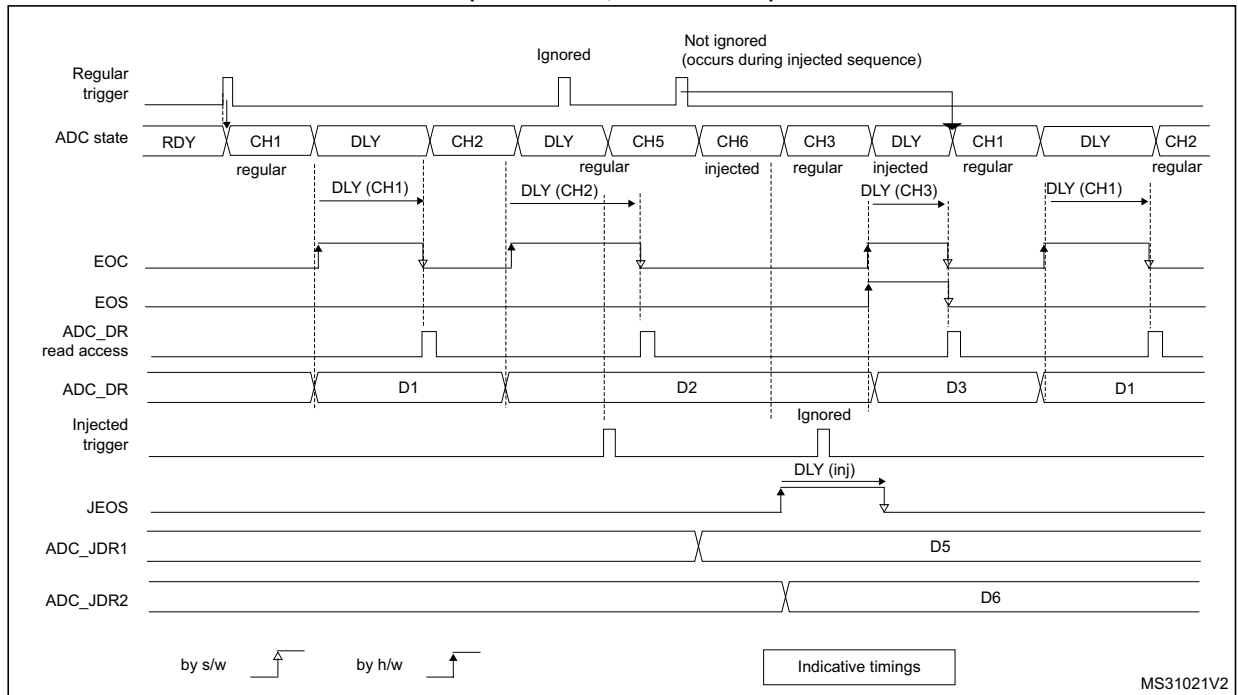
Figure 85. AUTDLY=1, regular conversion in continuous mode, software trigger



MS31020V1

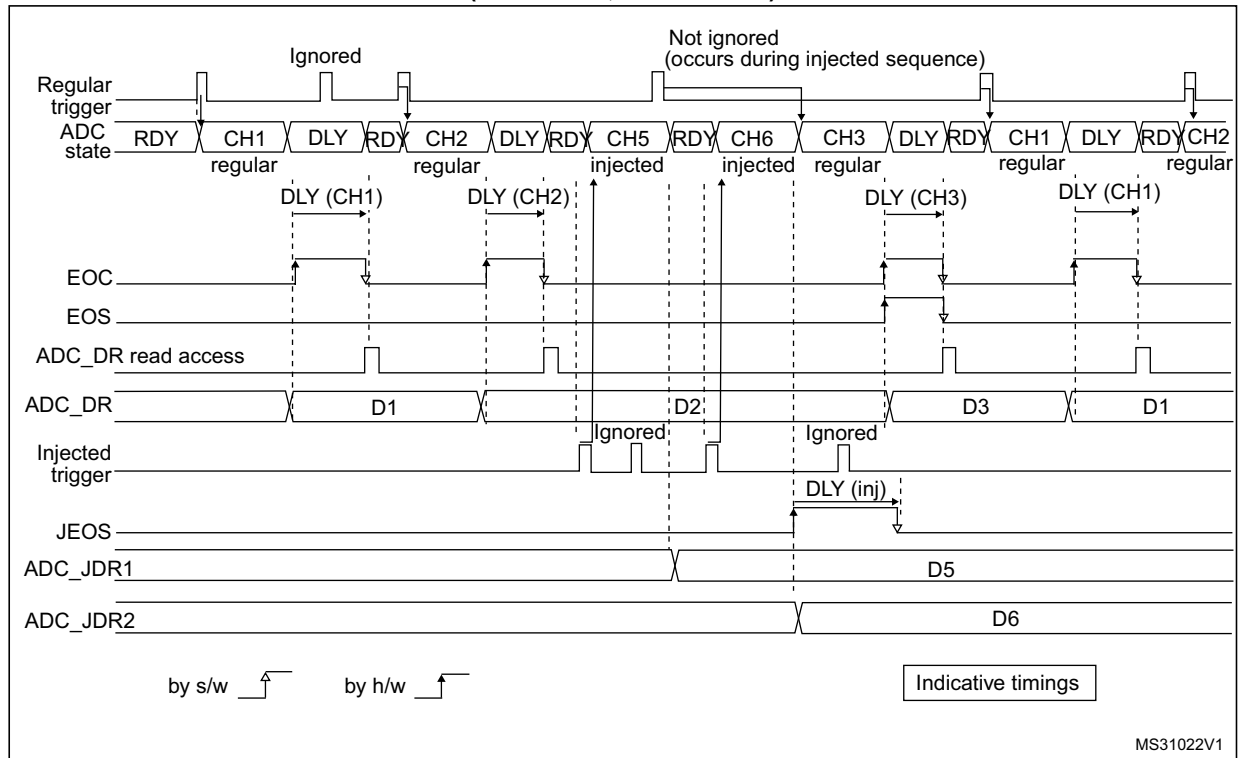
1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, CHANNELS = 1,2,3
3. Injected configuration DISABLED

Figure 86. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0)

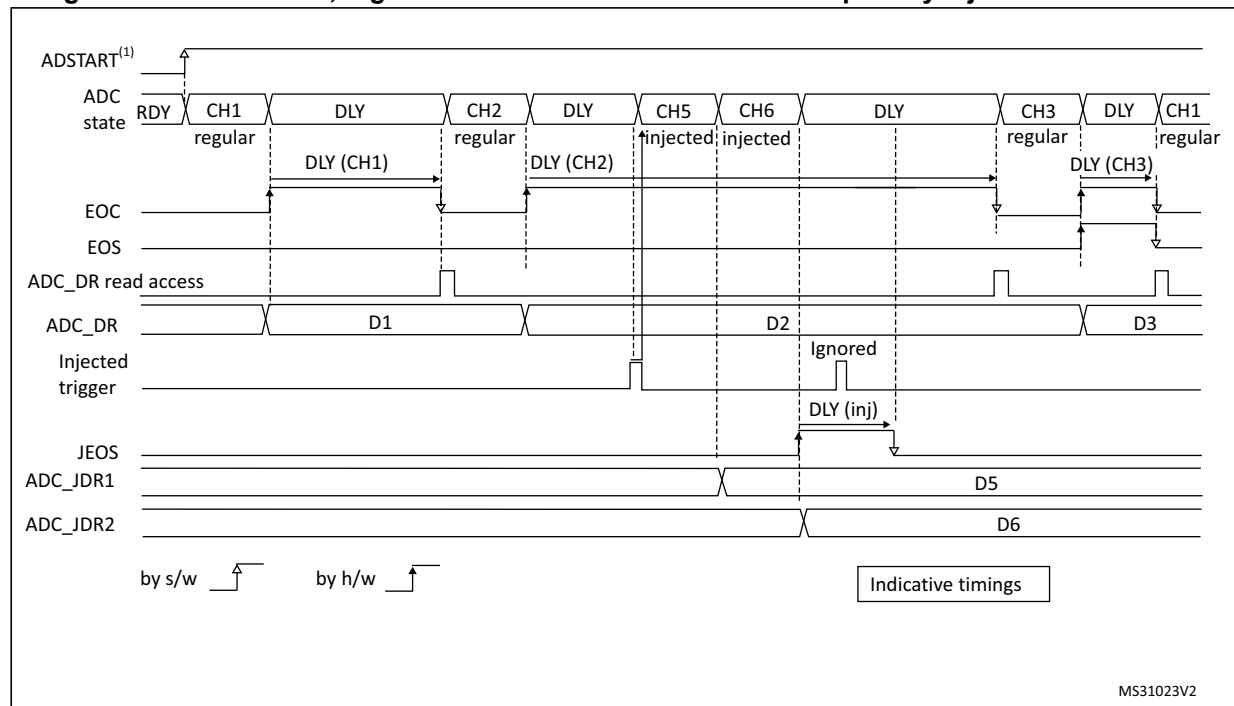


1. AUTDLY=1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

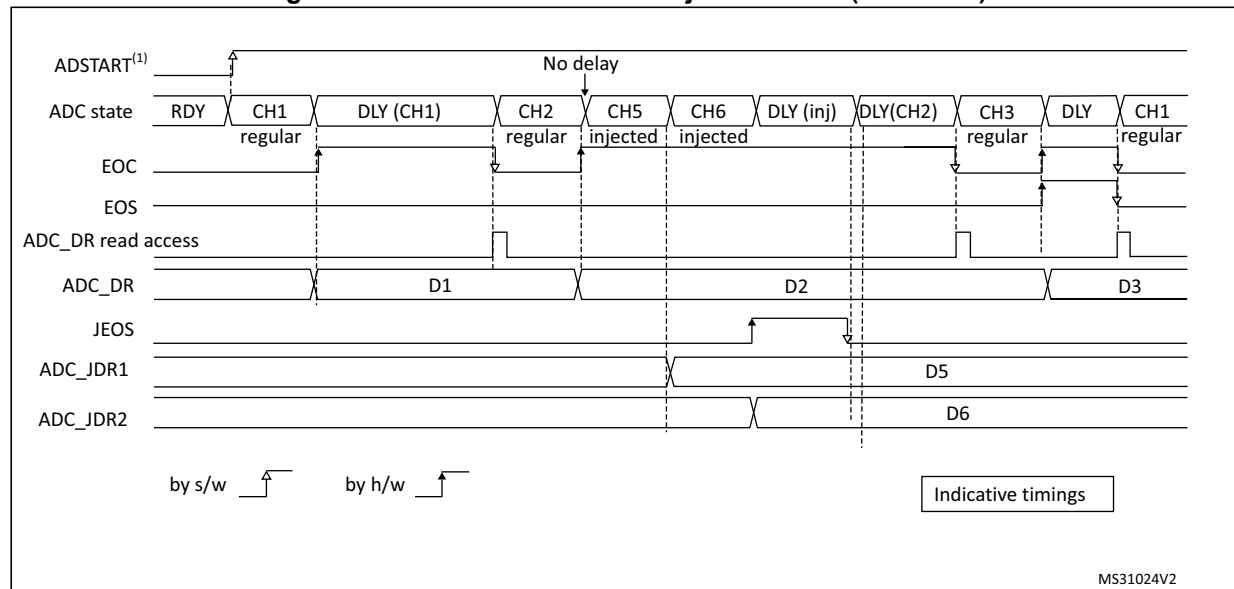
Figure 87. AUTDLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1)



1. AUTDLY=1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN=1, DISCNUM=1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=1, CHANNELS = 5,6

Figure 88. AUTODLY=1, regular continuous conversions interrupted by injected conversions

1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

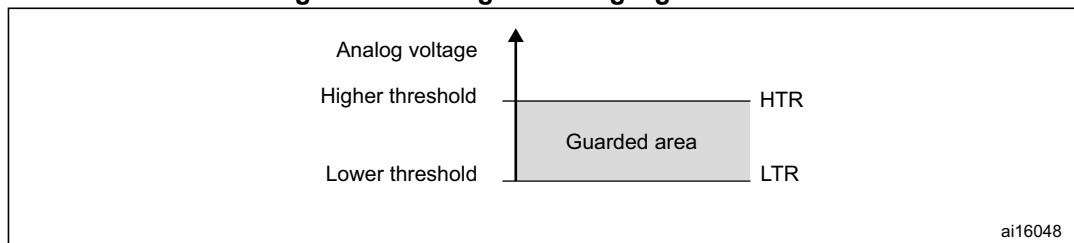
Figure 89. AUTODLY=1 in auto-injected mode (JAUTO=1)

1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2
3. Injected configuration: JAUTO=1, CHANNELS = 5,6

15.3.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 90. Analog watchdog's guarded area



ai16048

AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDxIE in the ADCx_IER register (x=1,2,3).

AWDx (x=1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADCx_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels⁽¹⁾ remain within a configured voltage range (window).

[Table 90](#) shows how the ADCx_CFGR registers should be configured to enable the analog watchdog on one or more channels.

Table 90. Analog watchdog channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single ⁽¹⁾ injected channel	1	0	1
Single ⁽¹⁾ regular channel	1	1	0
Single ⁽¹⁾ regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed in bits HT1[11:0] and LT1[11:0] of the ADCx_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits (according to bits RES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

[Table 91](#) describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 91. Analog watchdog 1 comparison

Resolution (bit RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned ⁽¹⁾	Thresholds	
00: 12-bit	DATA[11:0]	LT1[11:0] and HT1[11:0]	-
01: 10-bit	DATA[11:2],00	LT1[11:0] and HT1[11:0]	User must configure LT1[1:0] and HT1[1:0] to 00
10: 8-bit	DATA[11:4],0000	LT1[11:0] and HT1[11:0]	User must configure LT1[3:0] and HT1[3:0] to 0000
11: 6-bit	DATA[11:6],0000 00	LT1[11:0] and HT1[11:0]	User must configure LT1[5:0] and HT1[5:0] to 000000

1. The watchdog comparison is performed on the raw converted data before any alignment calculation and before applying any offsets (the data which is compared is not signed).

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCHx[19:0] (x=2,3).

The corresponding watchdog is enabled when any bit of AWDCHx[19:0] (x=2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. [Table 92](#) describes how the comparison is performed for all the possible resolutions.

Table 92. Analog watchdog 2 and 3 comparison

Resolution (bits RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned ⁽¹⁾	Thresholds	
00: 12-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:0] are not relevant for the comparison
01: 10-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:2] are not relevant for the comparison
10: 8-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	-
11: 6-bit	DATA[11:6],00	LTx[7:0] and HTx[7:0]	User must configure LTx[1:0] and HTx[1:0] to 00

1. The watchdog comparison is performed on the raw converted data before any alignment calculation and before applying any offsets (the data which is compared is not signed).

ADCy_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCy_AWDx_OUT (y=ADC number, x=watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCy_AWDx_OUT signal as ETR.

ADCy_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADCy_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCy_AWDx_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADCy_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS=1).
Note that stopping regular or injected conversions (setting ADSTP=1 or JADSTP=1) has no influence on the generation of ADCy_AWDx_OUT.

Note: AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADCy_AWDx_OUT (ex: ADCy_AWDx_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Figure 91. ADCy_AWDx_OUT signal generation (on all regular channels)

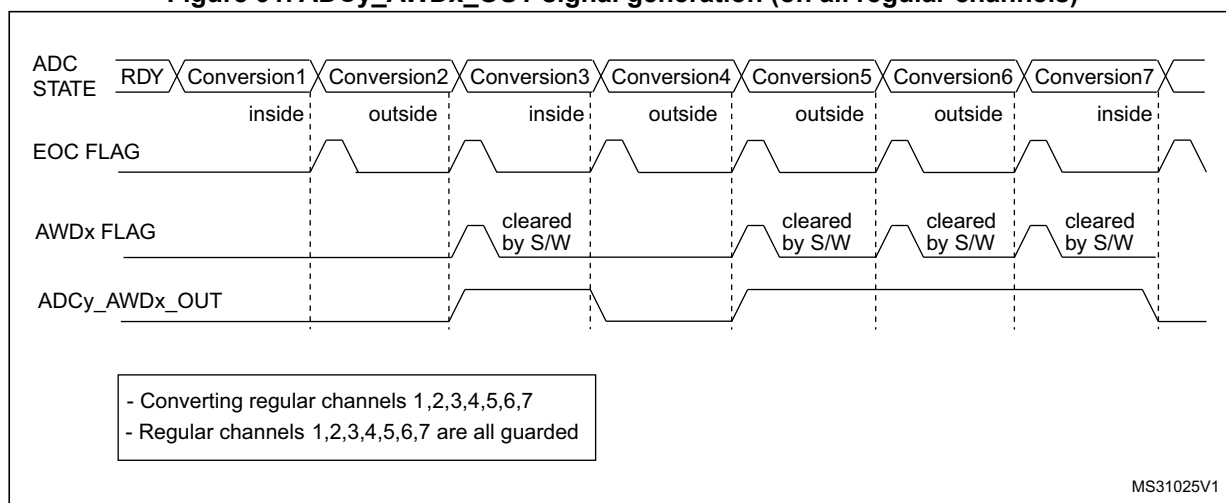


Figure 92. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by SW)

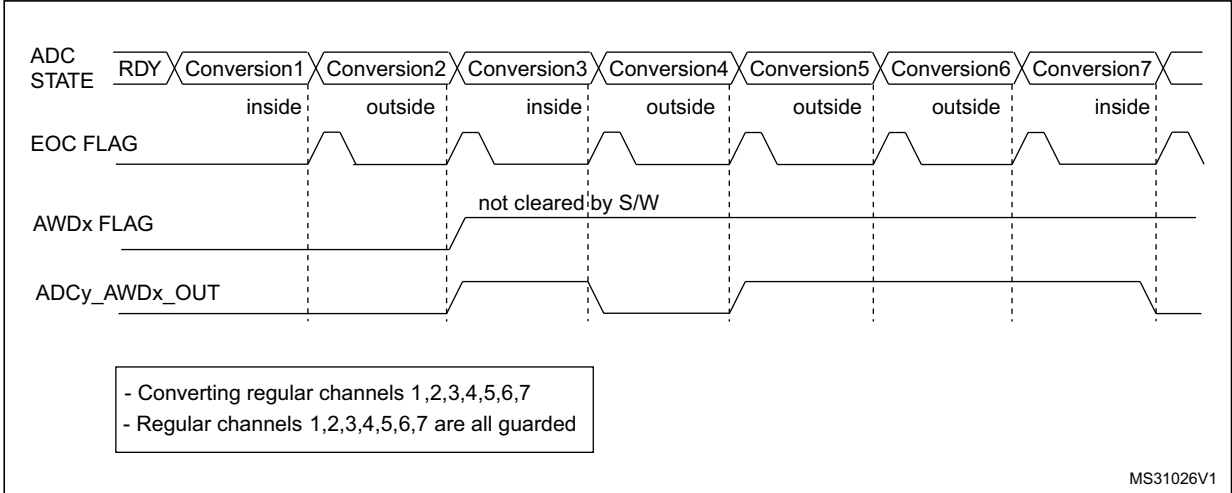


Figure 93. ADCy_AWDx_OUT signal generation (on a single regular channel)

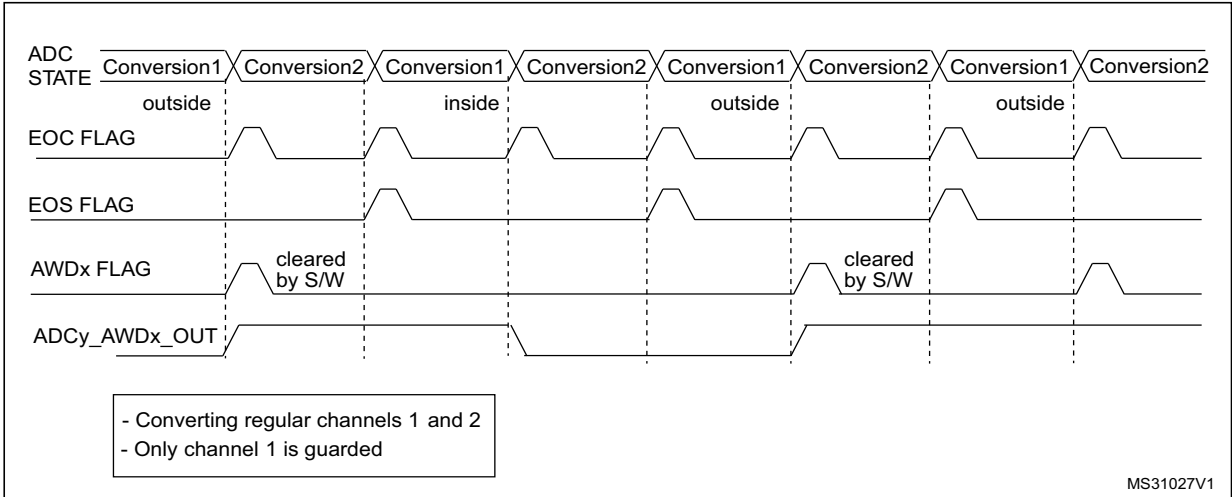
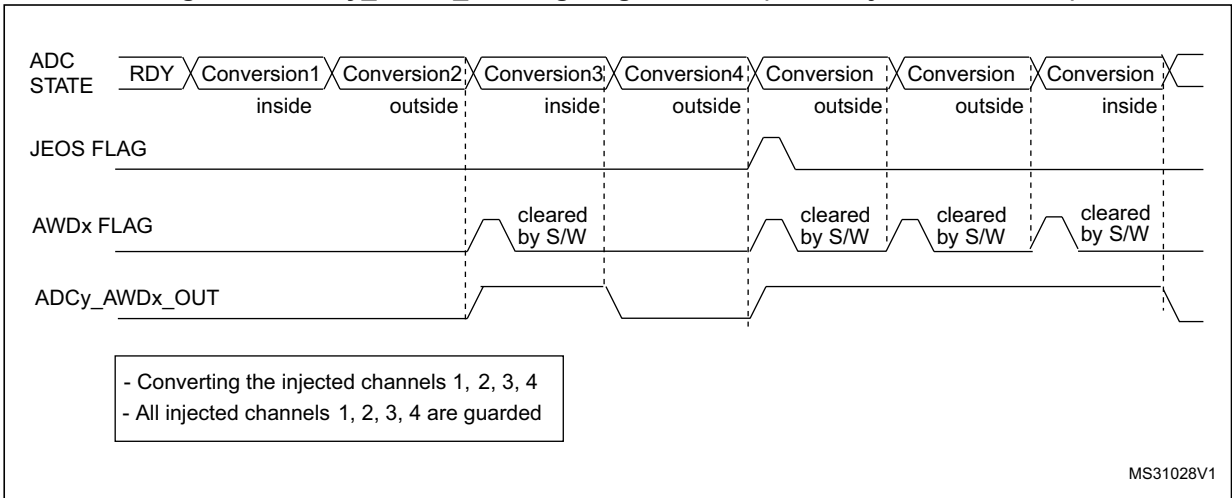


Figure 94. ADCy_AWDx_OUT signal generation (on all injected channels)



15.3.29 Dual ADC modes (STM32F302xB/C only)

In devices with two ADCs or more, dual ADC modes can be used (see [Figure 95](#)):

- ADC1 and ADC2 can be used together in dual mode (ADC1 is master)

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADCx master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADCx_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode

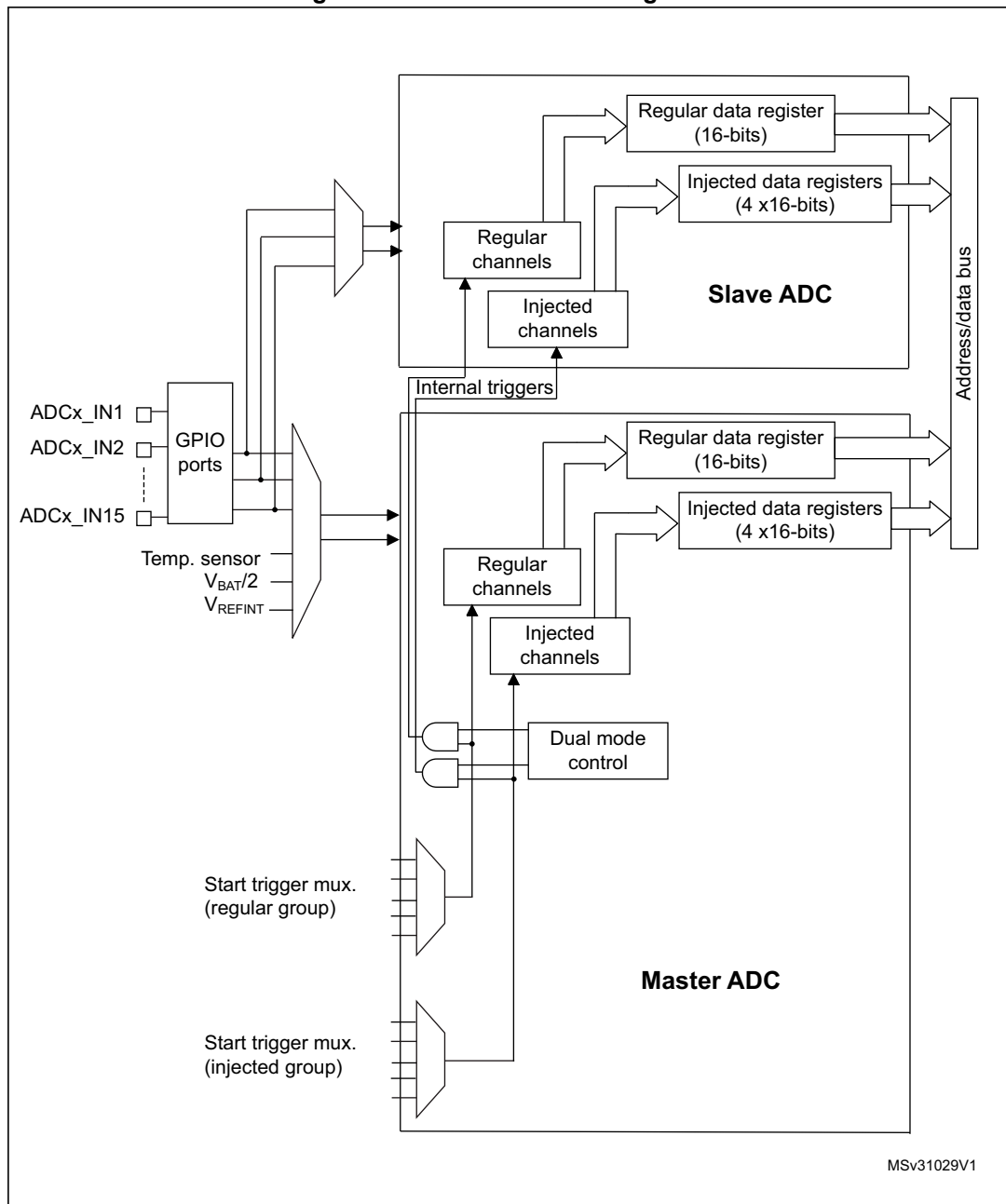
In dual ADC mode (when bits DUAL[4:0] in ADCx_CCR register are not equal to zero), the bits CONT, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JQM, JAUTO of the ADCx_CFGR register are shared between the master and slave ADC: the bits in the slave ADC are always equal to the corresponding bits of the master ADC.

To start a conversion in dual mode, the user must program the bits EXTEN, EXTSEL, JEXTEN, JEXTSEL of the master ADC only, to configure a software or hardware trigger, and a regular or injected trigger. (the bits EXTEN[1:0] and JEXTEN[1:0] of the slave ADC are don't care).

In regular simultaneous or interleaved modes: once the user sets bit ADSTART or bit ADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit ADSTART or bit ADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes: once the user sets bit JADSTART or bit JADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit JADSTART or bit JADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADCx_CDR). The status bits can be also read in parallel by reading the dual-mode status register (ADCx_CSR).

Figure 95. Dual ADC block diagram⁽¹⁾

1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADCx_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode

This mode is selected by programming bits DUAL[4:0]=00101

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL[3:0] bits in the ADCx_JSQR register).

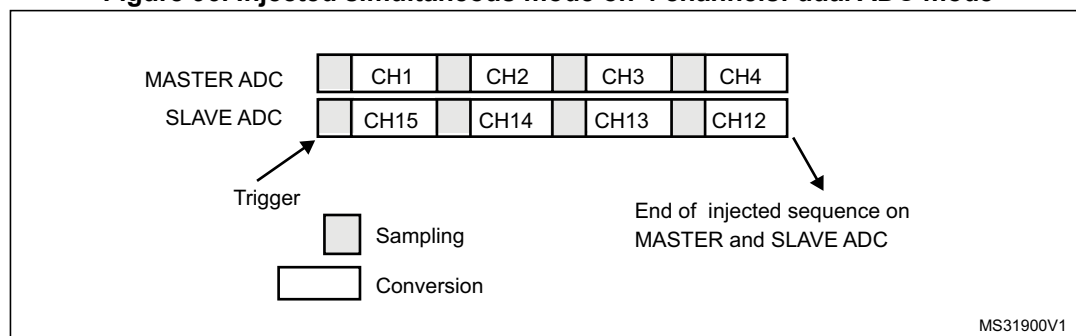
Note: *Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).*

In simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADCx_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADCx_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master injected sequence is equal to the duration of the slave injected one (like in [Figure 96](#)), it is possible for the software to enable only one of the two JEOS interrupt (ex: master JEOS) and read both converted data (from master ADCx_JDRy and slave ADCx_JDRy registers).

Figure 96. Injected simultaneous mode on 4 channels: dual ADC mode



If JDISCEN=1, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADCx_DR) has been read. Any new regular trigger events occurring for the master ADC during the

ongoing regular sequence and the associated delay phases are ignored.
There is the same behavior for regular sequences occurring on the slave ADC.

Regular simultaneous mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00110.

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL[3:0] bits in the ADCx_CFGR register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on master or on the slave) will abort the current simultaneous conversions, which are re-started once the injected conversion is completed.

Note: *Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).*
In regular simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Software is notified by interrupts when it can read the data:

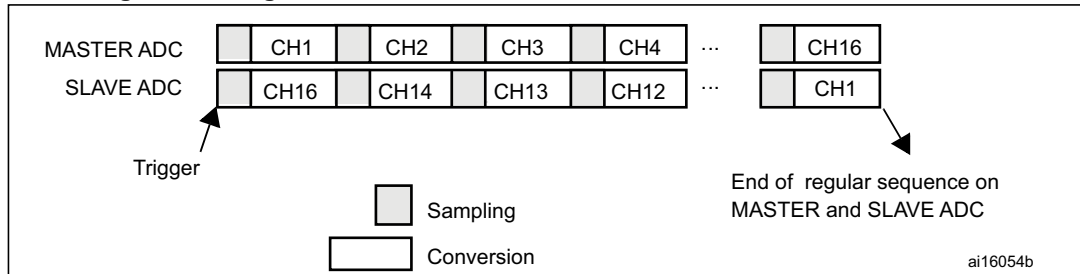
- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADCx_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADCx_DR of the slave ADC.
- If the duration of the master regular sequence is equal to the duration of the slave one (like in [Figure 97](#)), it is possible for the software to enable only one of the two EOC interrupt (ex: master EOC) and read both converted data from the Common Data register (ADCx_CDR).

It is also possible to read the regular data using the DMA. Two methods are possible:

- Using two DMA channels (one for the master and one for the slave). In this case bits MDMA[1:0] must be kept cleared.
 - Configure the DMA master ADC channel to read ADCx_DR from the master. DMA requests are generated at each EOC event of the master ADC.
 - Configure the DMA slave ADC channel to read ADCx_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Using MDMA mode, which leaves one DMA channel free for other uses:
 - Configure MDMA[1:0]=0b10 or 0b11 (depending on resolution).
 - A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR)
 - A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
 - both EOC flags are cleared when the DMA reads the ADCx_CCR register.

Note: In MDMA mode ($MDMA[1:0]=0b10$ or $0b11$), the user must program the same number of conversions in the master's sequence as in the slave's sequence. Otherwise, the remaining conversions will not generate a DMA request.

Figure 97. Regular simultaneous mode on 16 channels: dual ADC mode



If $DISCEN=1$ then each “n” simultaneous conversions of the regular sequence require a regular trigger event to occur (“n” is defined by $DISCNUM$).

This mode can be combined with AUTDLY mode:

- Once a simultaneous conversion of the sequence has ended, the next conversion in the sequence is started only if the common data register, $ADCx_CDR$ (or the regular data register of the master ADC) has been read (delay phase).
- Once a simultaneous regular sequence of conversions has ended, a new regular trigger event is accepted only if the common data register ($ADCx_CDR$) has been read (delay phase). Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

It is possible to use the DMA to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multi-DMA mode is used: bits $MDMA$ must be set to $0b10$ or $0b11$.

When regular simultaneous mode is combined with AUTDLY mode, it is mandatory for the user to ensure that:

- The number of conversions in the master's sequence is equal to the number of conversions in the slave's.
- For each simultaneous conversions of the sequence, the length of the conversion of the slave ADC is inferior to the length of the conversion of the master ADC. Note that the length of the sequence depends on the number of channels to convert and the sampling time and the resolution of each channels.

Note: This combination of regular simultaneous mode and AUTDLY mode is restricted to the use case when only regular channels are programmed: it is forbidden to program injected channels in this combined mode.

Interleaved mode with independent injected

This mode is selected by programming bits $DUAL[4:0] = 00111$.

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately.
- The slave ADC starts after a delay of several-ADC clock cycles after the sampling phase of the master ADC has complete.

The minimum delay which separates 2 conversions in interleaved mode is configured in the DELAY bits in the ADCx_CCR register. This delay starts to count after the end of the sampling phase of the master conversion. This way, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

If the CONT bit is set on both master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

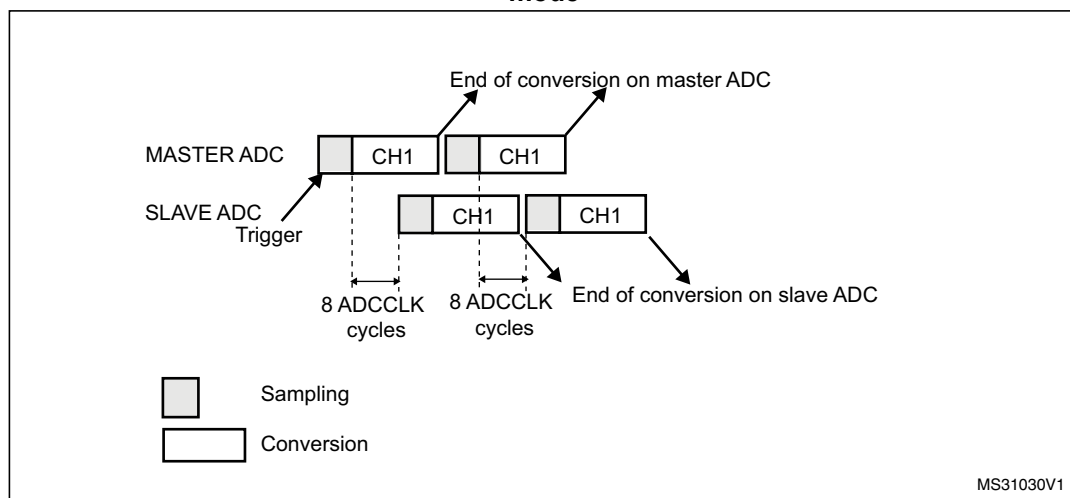
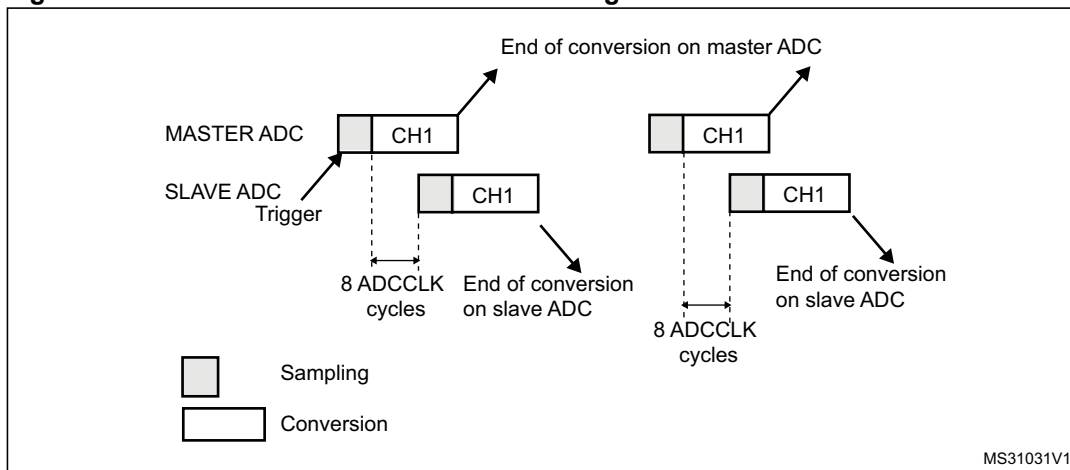
Software is notified by interrupts when it can read the data:

- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADCx_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADCx_DR of the slave ADC.

Note: *It is possible to enable only the EOC interrupt of the slave and read the common data register (ADCx_CDR). But in this case, the user must ensure that the duration of the conversions are compatible to ensure that inside the sequence, a master conversion is always followed by a slave conversion before a new master conversion restarts.*

It is also possible to read the regular data using the DMA. Two methods are possible:

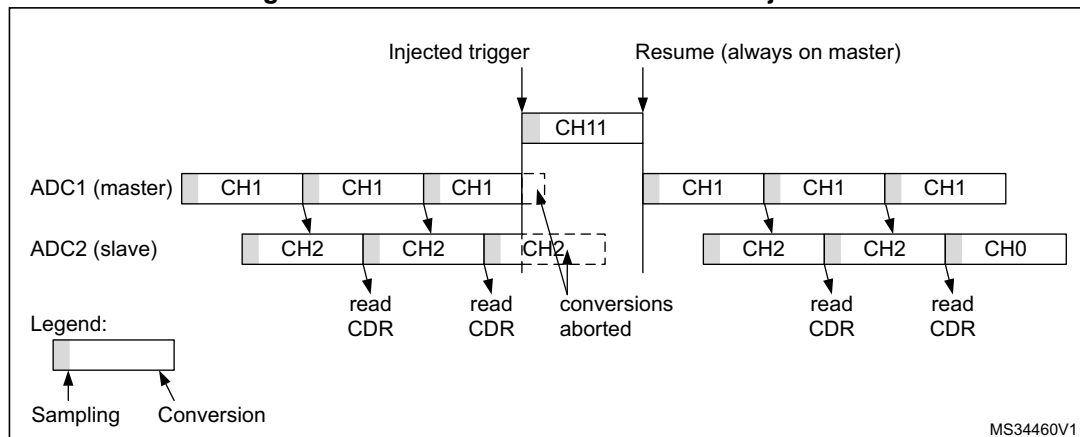
- Using the two DMA channels (one for the master and one for the slave). In this case bits MDMA[1:0] must be kept cleared.
 - Configure the DMA master ADC channel to read ADCx_DR from the master. DMA requests are generated at each EOC event of the master ADC.
 - Configure the DMA slave ADC channel to read ADCx_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Using MDMA mode, which allows to save one DMA channel:
 - Configure MDMA[1:0]=0b10 or 0b11 (depending on resolution).
 - A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR).
 - A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
 - Both EOC flags are cleared when the DMA reads the ADCx_CCR register.

Figure 98. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode**Figure 99. Interleaved mode on 1 channel in single conversion mode: dual ADC mode**

If DISCEN=1, each “n” simultaneous conversions (“n” is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In this mode, injected conversions are supported. When injection is done (either on master or on slave), both the master and the slave regular conversions are aborted and the sequence is re-started from the master (see [Figure 100](#) below).

Figure 100. Interleaved conversion with injection



Alternate trigger mode

This mode is selected by programming bits DUAL[4:0] = 01001.

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN must not be 0x0.

Injected discontinuous mode disabled (JDISCEN=0 for both ADC)

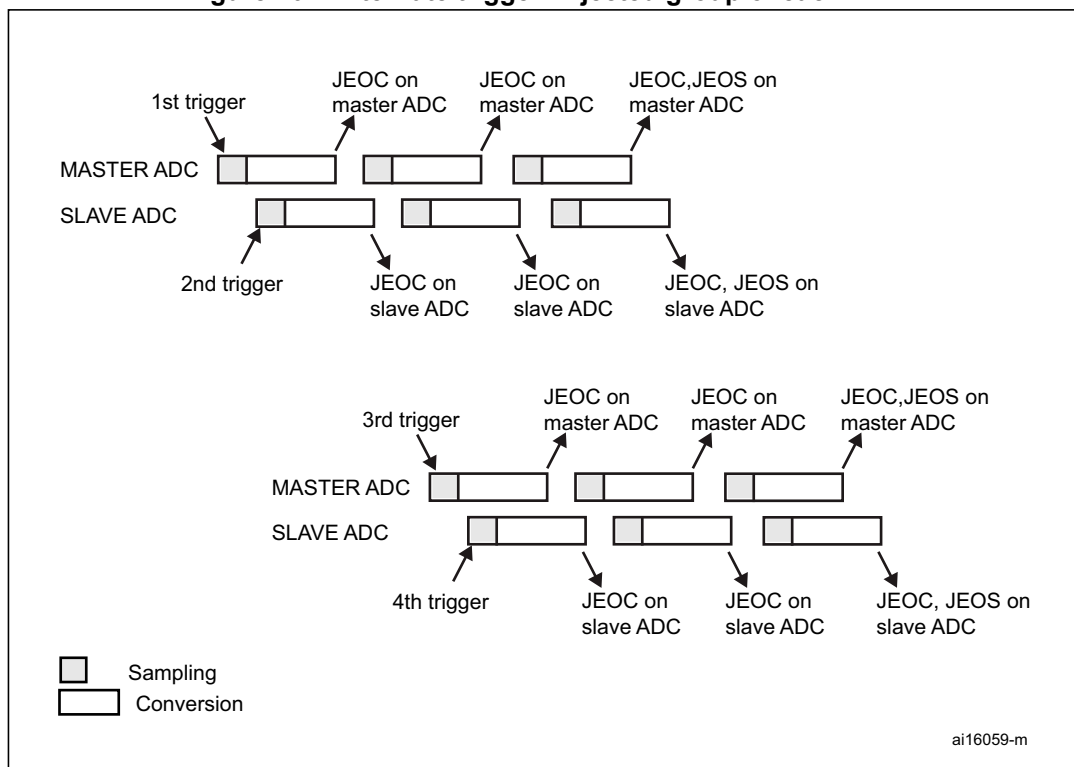
1. When the 1st trigger occurs, all injected master ADC channels in the group are converted.
2. When the 2nd trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on.

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOS interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected channels of the master ADC in the group.

Figure 101. Alternate trigger: injected group of each ADC

Note: Regular conversions can be enabled on one or all ADCs. In this case the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between 2 trigger events must be greater than or equal to 1 ADC clock period. The minimum time interval between 2 trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN=1 for both ADC)

If the injected discontinuous mode is enabled for both master and slave ADCs:

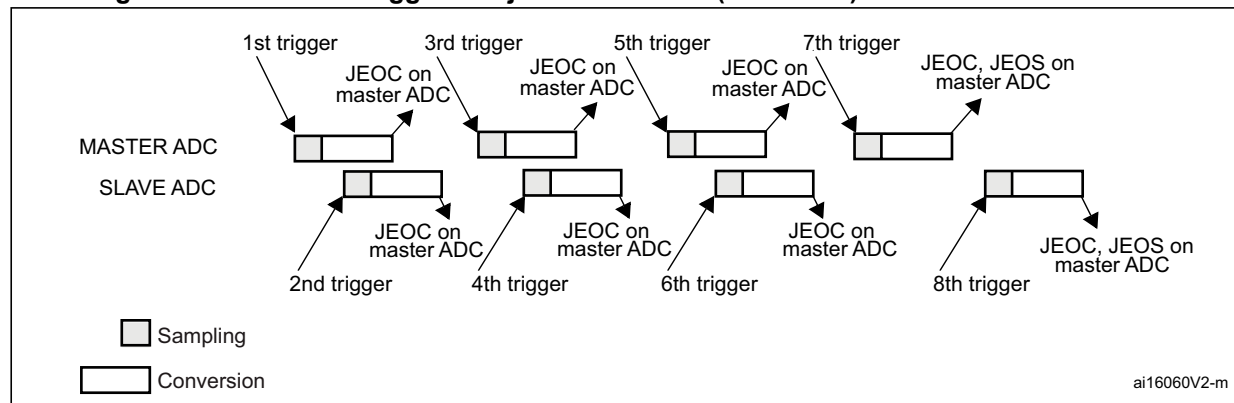
- When the 1st trigger occurs, the first injected channel of the master ADC is converted.
- When the 2nd trigger occurs, the first injected channel of the slave ADC is converted.
- And so on.

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversions.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

Figure 102. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode

Combined regular/injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00001.

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

Note: *In combined regular/injected simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.*

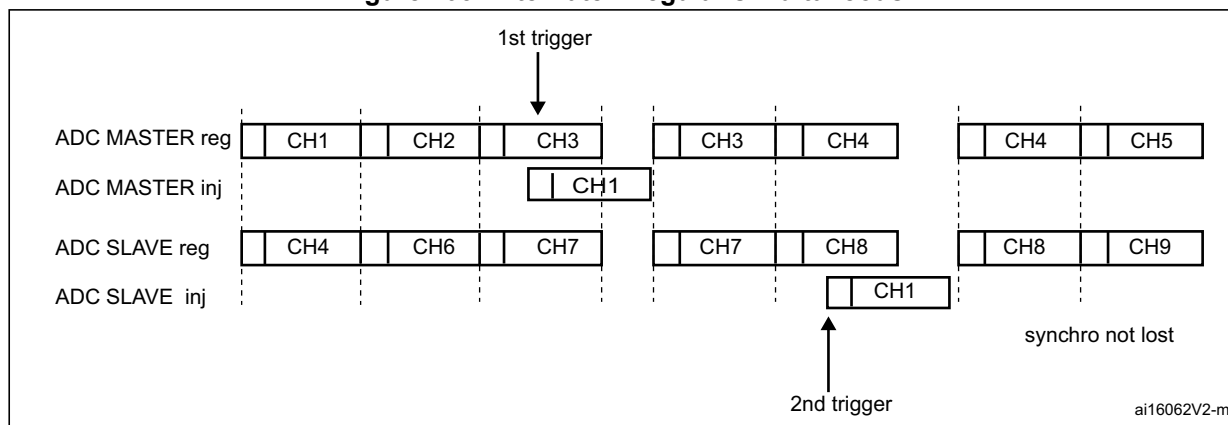
Combined regular simultaneous + alternate trigger mode

This mode is selected by programming bits DUAL[4:0]=00010.

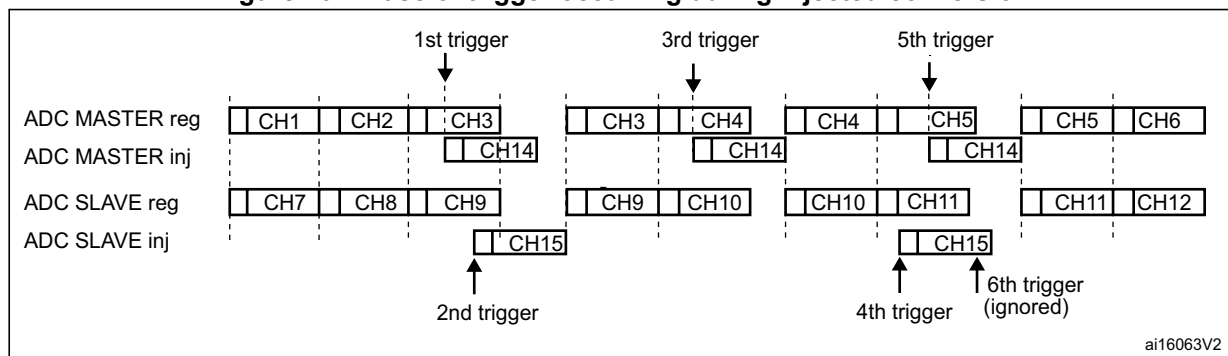
It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. [Figure 103](#) shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

Note: *In combined regular simultaneous + alternate trigger mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.*

Figure 103. Alternate + regular simultaneous

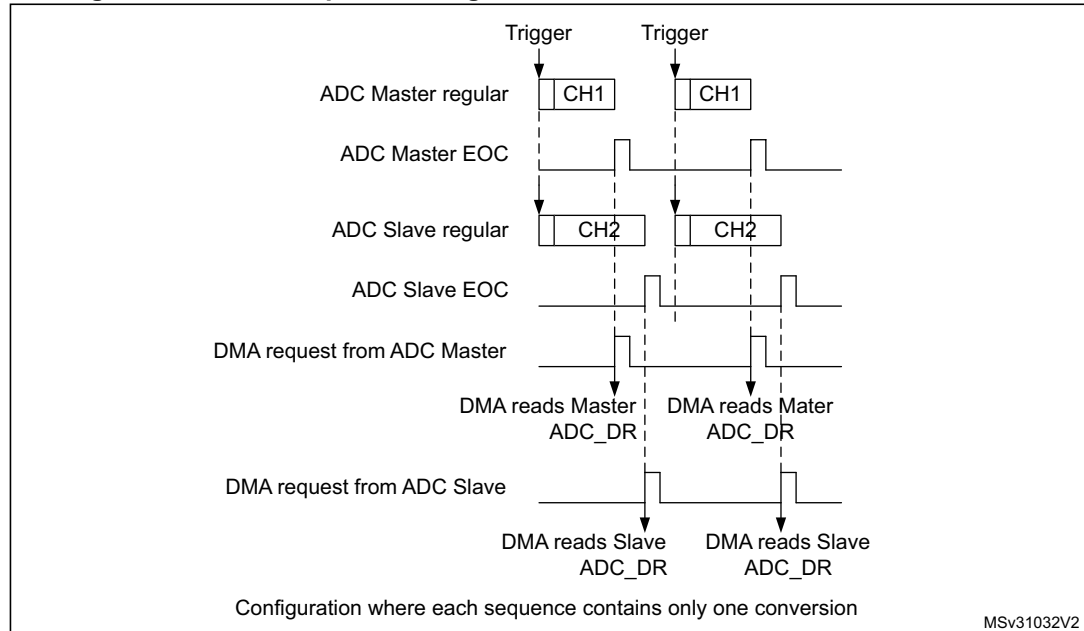
If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. [Figure 104](#) shows the behavior in this case (note that the 6th trigger is ignored because the associated alternate conversion is not complete).

Figure 104. Case of trigger occurring during injected conversion

DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to [Figure 105: DMA Requests in regular simultaneous mode when MDMA=0b00](#)).

Figure 105. DMA Requests in regular simultaneous mode when MDMA=0b00



In simultaneous regular and interleaved modes, it is also possible to save one DMA channel and transfer both data using a single DMA channel. For this MDMA bits must be configured in the ADCx_CCR register:

- **MDMA=0b10:** A single DMA request is generated each time both master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADCx_CDR contains the two half-words representing two ADC-converted data items. The slave ADC data take the upper half-word and the master ADC data take the lower half-word.

This mode is used in interleaved mode and in regular simultaneous mode when resolution is 10-bit or 12-bit.

Example:

Interleaved dual mode: a DMA request is generated each time 2 data items are available:

1st DMA request: ADCx_CDR[31:0] = SLV_ADCx_DR[15:0] | MST_ADCx_DR[15:0]

2nd DMA request: ADCx_CDR[31:0] = SLV_ADCx_DR[15:0] | MST_ADCx_DR[15:0]

Figure 106. DMA requests in regular simultaneous mode when MDMA=0b10

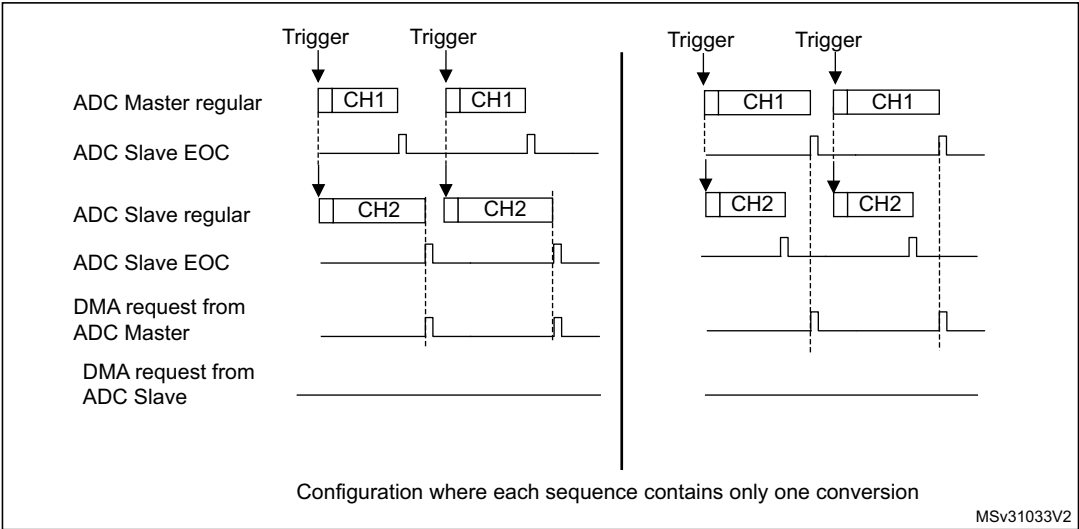
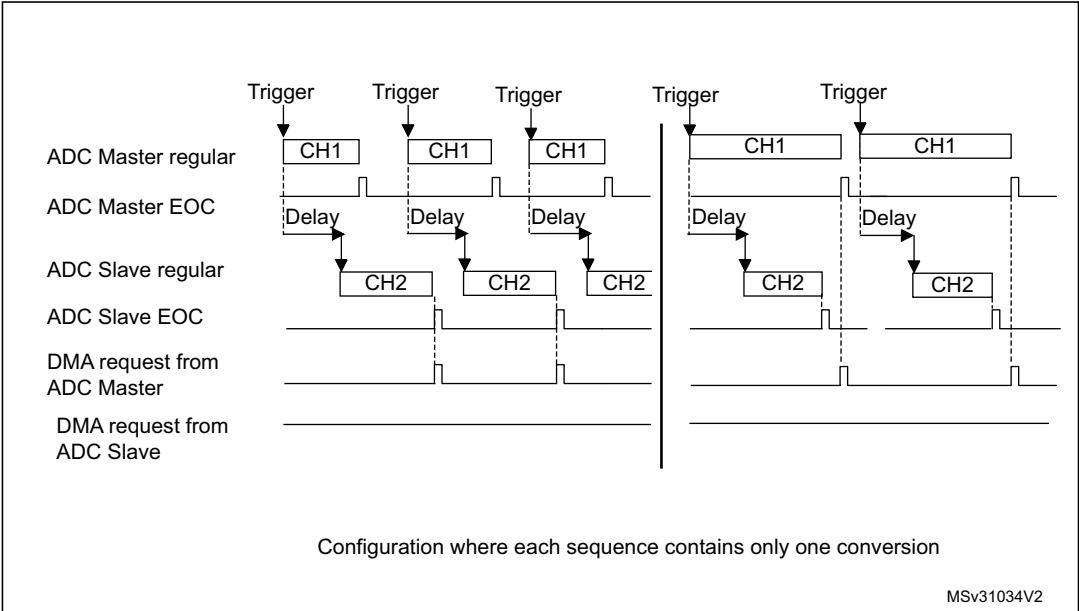


Figure 107. DMA requests in interleaved mode when MDMA=0b10



Note: When using MDMA mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion is available.

- **MDMA=0b11:** This mode is similar to the MDMA=0b10. The only differences are that on each DMA request (two data items are available), two bytes representing two ADC converted data items are transferred as a half-word.

This mode is used in interleaved and regular simultaneous mode when resolution is 6-bit or when resolution is 8-bit and data is not signed (offsets must be disabled for all the involved channels).

Example:

Interleaved dual mode: a DMA request is generated each time 2 data items are available:

1st DMA request: `ADCx_CDR[15:0] = SLV_ADCx_DR[7:0] | MST_ADCx_DR[7:0]`

2nd DMA request: `ADCx_CDR[15:0] = SLV_ADCx_DR[7:0] | MST_ADCx_DR[7:0]`

Overrun detection

In dual ADC mode (when `DUAL[4:0]` is not equal to `b00000`), if an overrun is detected on one of the ADCs, the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the MDMA configuration). It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

DMA one shot mode/ DMA circular mode when MDMA mode is selected

When MDMA mode is selected (0b10 or 0b11), bit `DMACFG` of the `ADCx_CCR` register must also be configured to select between DMA one shot mode and circular mode, as explained in section [Section : Managing conversions using the DMA](#) (bits `DMACFG` of master and slave `ADCx_CFGR` are not relevant).

Stopping the conversions in dual ADC modes

The user must set the control bits `ADSTP/JADSTP` of the master ADC to stop the conversions of both ADC in dual ADC mode. The other `ADSTP` control bit of the slave ADC has no effect in dual ADC mode.

Once both ADC are effectively stopped, the bits `ADSTART/JADSTART` of the master and slave ADCs are both cleared by hardware.

15.3.30 Temperature sensor

The temperature sensor can be used to measure the junction temperature (T_J) of the device. The temperature sensor is internally connected to the input channels which are used to convert the sensor output voltage to a digital value. When not in use, the sensor can be put in power down mode.

[Figure 108](#) shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the

temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference. Refer to the STM32F302xx datasheet for additional information.

Main features

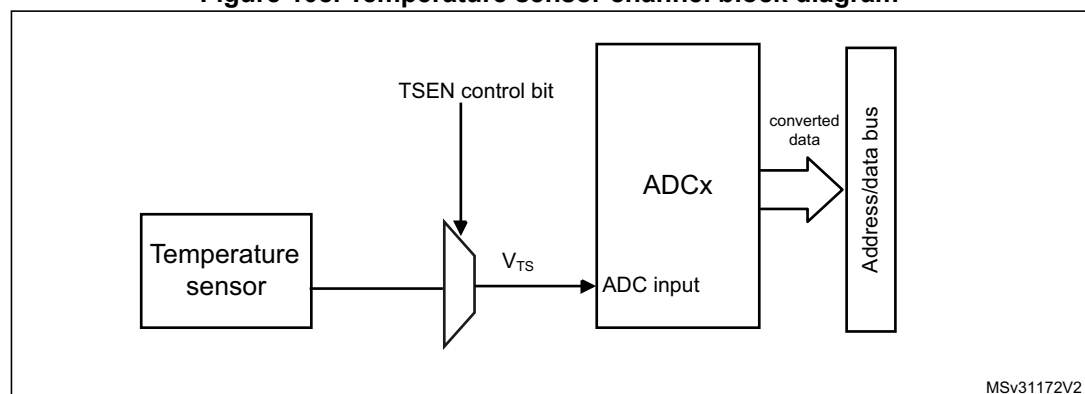
- Supported temperature range: -40 to 125 °C
- Precision: ± 2 °C

The temperature sensor is internally connected to the ADC1_IN16 input channel which is used to convert the sensor's output voltage to a digital value. Refer to the electrical characteristics section of STM32F302xx datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

Figure 108 shows the block diagram of the temperature sensor.

Figure 108. Temperature sensor channel block diagram



Note: The TSEN bit must be set to enable the conversion of the temperature sensor voltage V_{TS} .

Reading the temperature

To use the sensor:

1. Select the ADC1_IN16 input channel (with the appropriate sampling time).
2. Program with the appropriate sampling time (refer to electrical characteristics section of the STM32F302xx datasheet).
3. Set the TSEN bit in the ADC1_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V_{TS} data in the ADC data register.
6. Calculate the actual temperature using the following formula:

$$\text{Temperature (in } ^\circ\text{C)} = \{(V_{25} - V_{TS}) / \text{Avg_Slope}\} + 25$$

Where:

- $V_{25} = V_{TS}$ value for 25°C
- Avg_Slope = average slope of the temperature vs. V_{TS} curve (given in mV/ $^\circ\text{C}$ or $\mu\text{V}/^\circ\text{C}$)

Refer to the datasheet electrical characteristics section for the actual values of V_{25} and Avg_Slope.

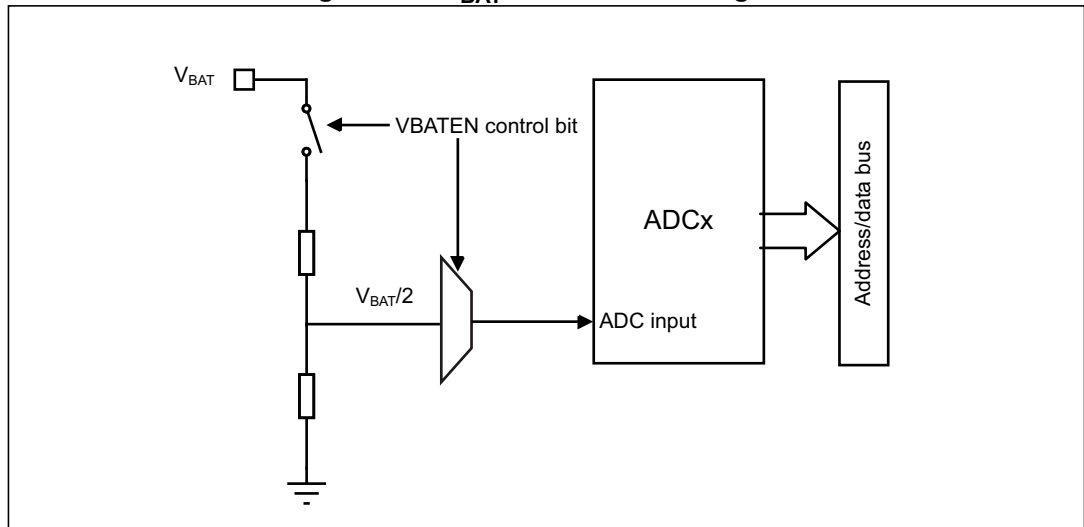
Note: *The sensor has a startup time after waking from power-down mode before it can output V_{TS} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and TSEN bits should be set at the same time.*

15.3.31 V_{BAT} supply monitoring

The VBATEN bit in the ADC12_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage could be higher than V_{DDA} , to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider by 2. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/2$ to the ADC1_IN17 input channel. As a consequence, the converted digital value is half the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the STM32F302xx datasheet for the sampling time value to be applied when converting the $V_{BAT}/2$ voltage.

Figure 109 shows the block diagram of the V_{BAT} sensing feature.

Figure 109. V_{BAT} channel block diagram

Note: The $VBATEN$ bit must be set to enable the conversion of internal channel $ADC1_IN17$ (V_{BATEN}).

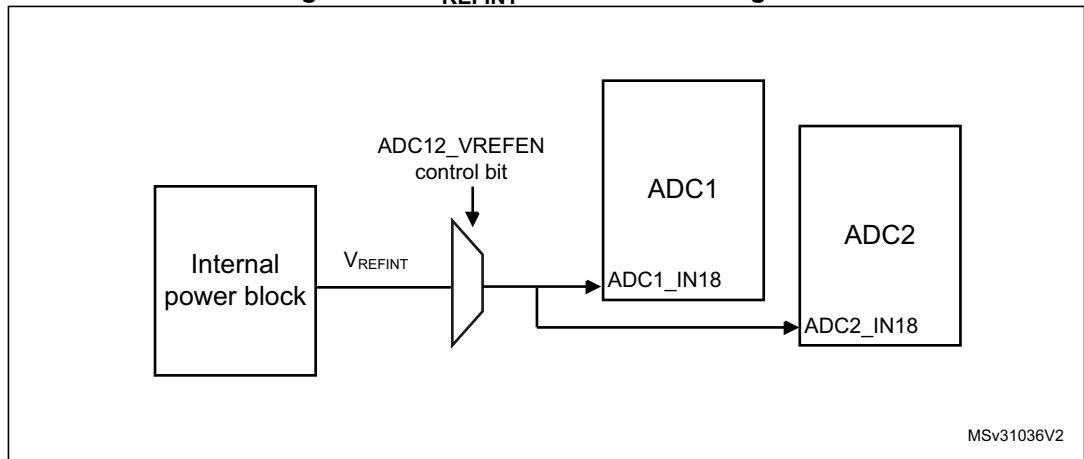
15.3.32 Monitoring the internal voltage reference

It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

The internal voltage reference is internally connected to the input channel 18 of the two ADCs ($ADCx_IN18$).

Refer to the electrical characteristics section of the STM32F302xx datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

Figure 109 shows the block diagram of the V_{REFINT} sensing feature.

Figure 110. V_{REFINT} channel block diagram

Note: The $VREFEN$ bit into $ADC12_CCR$ register must be set to enable the conversion of internal channels $ADC1_IN18$ or $ADC2_IN18$ (V_{REFINT}).

Calculating the actual V_{DDA} voltage using the internal reference voltage

The V_{DDA} power supply voltage applied to the microcontroller may be subject to variation or not precisely known. The embedded internal voltage reference (V_{REFINT}) and its calibration data acquired by the ADC during the manufacturing process at $V_{DDA} = 3.3$ V can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{DDA} voltage supplying the device:

$$V_{DDA} = 3.3 \text{ V} \times VREFINT_CAL / VREFINT_DATA$$

Where:

- $VREFINT_CAL$ is the $VREFINT$ calibration value
- $VREFINT_DATA$ is the actual $VREFINT$ output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between the analog power supply and the voltage applied on the converted channel. For most application use cases, it is necessary to convert this ratio into a voltage independent of V_{DDA} . For applications where V_{DDA} is known and ADC converted values are right-aligned you can use the following formula to get this absolute value:

$$V_{CHANNELx} = \frac{V_{DDA}}{FULL_SCALE} \times ADCx_DATA$$

For applications where V_{DDA} value is not known, you must use the internal voltage reference and V_{DDA} can be replaced by the expression provided in the section [Calculating the actual \$V_{DDA}\$ voltage using the internal reference voltage](#), resulting in the following formula:

$$V_{CHANNELx} = \frac{3.3 \text{ V} \times VREFINT_CAL \times ADCx_DATA}{VREFINT_DATA \times FULL_SCALE}$$

Where:

- $VREFINT_CAL$ is the $VREFINT$ calibration value
- $ADCx_DATA$ is the value measured by the ADC on channel x (right-aligned)
- $VREFINT_DATA$ is the actual $VREFINT$ output value converted by the ADC
- $FULL_SCALE$ is the maximum digital value of the ADC output. For example with 12-bit resolution, it will be $2^{12} - 1 = 4095$ or with 8-bit resolution, $2^8 - 1 = 255$.

Note: *If ADC measurements are done using an output format other than 12 bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.*

15.4 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (flag OVR)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

Table 93. ADC interrupts per each ADC

Interrupt event	Event flag	Enable control bit
ADC ready	ADRDY	ADRDYIE
End of conversion of a regular group	EOC	EOCIE
End of sequence of conversions of a regular group	EOS	EOSIE
End of conversion of a injected group	JEOC	JEOCIE
End of sequence of conversions of an injected group	JEOS	JEOSIE
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
End of sampling phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE
Injected context queue overflows	JQOVF	JQOVFIE

15.5 ADC registers (for each ADC)

Refer to [Section 2.1 on page 42](#) for a list of abbreviations used in register descriptions.

15.5.1 ADC interrupt and status register (ADCx_ISR, x=1..2)

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.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOS	OVR	EOS	EOC	EOSMP	ADRDY
					r_w1	r_w1	r_w1	r_w1	r_w1	r_w1	r_w1	r_w1	rc_w1	r_w1	r_w1

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

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to [Section 15.3.21: Queue of context for injected conversions](#) for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADCx_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADCx_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADCx_TR1 register. It is cleared by software writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS**: Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 JEOC: Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADCx_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADCx_JDRy register

- 0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)
- 1: Injected channel conversion complete

Bit 4 OVR: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.

- 0: No overrun occurred (or the flag event was already acknowledged and cleared by software)
- 1: Overrun has occurred

Bit 3 EOS: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

- 0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)
- 1: Regular Conversions sequence complete

Bit 2 EOC: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADCx_DR register. It is cleared by software writing 1 to it or by reading the ADCx_DR register

- 0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)
- 1: Regular channel conversion complete

Bit 1 EOSMP: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

- 0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)
- 1: End of sampling phase reached

Bit 0 ADRDY: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN=1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

- 0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)
- 1: ADC is ready to start conversion

15.5.2 ADC interrupt enable register (ADCx_IER, x=1..2)

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.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
					rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.

0: Injected Context Queue Overflow interrupt disabled

1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 5 JEOCIE: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 4 OVRIE: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 3 EOSIE: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 2 EOCIE: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 1 EOSMPIE: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 0 ADRDYIE: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

15.5.3 ADC control register (ADCx_CR, x=1..2)

Address offset: 0x08

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AD CAL	ADCA LDIF	ADVREGEN[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	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.	JAD STP	AD STP	JAD START	AD START	AD DIS	AD EN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

Note: Software is allowed to launch a calibration by setting ADCAL only when ADEN=0.

Note: Software is allowed to update the calibration factor by writing ADCx_CALFACT only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing)

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential inputs mode for the calibration.

0: Writing ADCAL will launch a calibration in Single-ended inputs Mode.

1: Writing ADCAL will launch a calibration in Differential inputs Mode.

Note: Software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 29:28 **ADVREGEN[1:0]**: ADC voltage regulator enable

These bits are set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

00: Intermediate state required when moving the ADC voltage regulator from the enabled to the disabled state or from the disabled to the enabled state.

01: ADC Voltage regulator enabled.

10: ADC Voltage regulator disabled (Reset state)

11: reserved

For more details about the ADC voltage regulator enable and disable sequences, refer to [Section 15.3.6: ADC voltage regulator \(ADVREGEN\)](#).

Note: The software can program this bit field only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 JADSTP: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: Software is allowed to set JADSTP only when JADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC)

Note: In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Bit 4 ADSTP: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: Software is allowed to set ADSTP only when ADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC)

Note: In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Note: In dual ADC regular simultaneous mode and interleaved mode, the bit ADSTP of the master ADC must be used to stop regular conversions. The other ADSTP bit is inactive.

Bit 3 JADSTART: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN, a conversion will start immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode when software trigger is selected (JEXTSEL=0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.

- in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: Software is allowed to set JADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC)

Note: In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 2 ADSTART: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN, a conversion will start immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode when software trigger is selected (EXTSEL=0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.
- in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

Note: 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)

Note: In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 1 ADDIS: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: Software is allowed to set ADDIS only when ADEN=1 and both ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

Bit 0 ADEN: ADC enable control

This bit is set by software to enable the ADC. The ADC will be effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: Software is allowed to set ADEN only when all bits of ADCx_CR registers are 0 (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)

15.5.4 ADC configuration register (ADCx_CFGR, x=1..2)

Address offset: 0x0C

Reset value: 0x0000 00000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWD1CH[4:0]					JAUTO	JAWD1 EN	AWD1 EN	AWD1S GL	JQM	JDISC EN	DISCNUM[2:0]			DISC EN
	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.	AUT DLY	CONT	OVR MOD	EXTEN[1:0]		EXTSEL[3:0]				ALIGN	RES[1:0]		Res.	DMA CFG	DMA EN
	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 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: reserved (analog input channel 0 is not mapped)

00001: ADC analog input channel-1 monitored by AWD1

.....

10010: ADC analog input channel-18 monitored by AWD1

others: reserved, must not be used

Note: The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO**: Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no regular nor injected conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit JAUTO of the slave ADC is no more writable and its content is equal to the bit JAUTO of the master ADC.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 21 **JQM**: JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: JSQR Mode 0: The Queue is never empty and maintains the last written configuration into JSQR.

1: JSQR Mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to [Section 15.3.21: Queue of context for injected conversions](#) for more information.

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit JQM of the slave ADC is no more writable and its content is equal to the bit JQM of the master ADC.

Bit 20 **JDISCEN**: Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: Software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Note: It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit JDISCEN of the slave ADC is no more writable and its content is equal to the bit JDISCEN of the master ADC.

Bits 19:17 **DISCNUM[2:0]**: Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bits DISCNUM[2:0] of the slave ADC are no more writable and their content is equal to the bits DISCNUM[2:0] of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable Discontinuous mode for regular channels.

- 0: Discontinuous mode for regular channels disabled
- 1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

Note: It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit DISCEN of the slave ADC is no more writable and its content is equal to the bit DISCEN of the master ADC.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode:

- 0: Auto-delayed conversion mode off
- 1: Auto-delayed conversion mode on

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit AUTDLY of the slave ADC is no more writable and its content is equal to the bit AUTDLY of the master ADC.

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

- 0: Single conversion mode
- 1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (bits DUAL of ADCx_CCR register are not equal to zero), the bit CONT of the slave ADC is no more writable and its content is equal to the bit CONT of the master ADC.

Bit 12 **OVRMOD**: Overrun Mode

This bit is set and cleared by software and configure the way data overrun is managed.

- 0: ADCx_DR register is preserved with the old data when an overrun is detected.
- 1: ADCx_DR register is overwritten with the last conversion result when an overrun is detected.

Note: Software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 9:6 **EXTSEL[3:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

0000: Event 0

0001: Event 1

0010: Event 2

0011: Event 3

0100: Event 4

0101: Event 5

0110: Event 6

0111: Event 7

...

1111: Event 15

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to [Figure : Data register, data alignment and offset \(ADCx_DR, OFFSETy, OFFSETy_CH, ALIGN\)](#)

0: Right alignment

1: Left alignment

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 4:3 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit

01: 10-bit

10: 8-bit

11: 6-bit

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMACFG**: Direct memory access configuration

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN=1.

0: DMA One Shot Mode selected

1: DMA Circular Mode selected

For more details, refer to [Section : Managing conversions using the DMA](#)

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Note: In dual-ADC modes, this bit is not relevant and replaced by control bit DMACFG of the ADCx_CCR register.

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the GP-DMA to manage automatically the converted data. For more details, refer to [Section : Managing conversions using the DMA](#).

0: DMA disabled

1: DMA enabled

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Note: In dual-ADC modes, this bit is not relevant and replaced by control bits MDMA[1:0] of the ADCx_CCR register.

15.5.5 ADC sample time register 1 (ADCx_SMPR1, x=1..2)

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.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			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	r/w			

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:3 **SMPx[2:0]**: Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles

001: 2.5 ADC clock cycles

010: 4.5 ADC clock cycles

011: 7.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 61.5 ADC clock cycles

110: 181.5 ADC clock cycles

111: 601.5 ADC clock cycles

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 2:0 Reserved

15.5.6 ADC sample time register 2 (ADCx_SMPR2, x=1..2)

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.	SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
					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
SMP15_0		SMP14[2:0]		SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bits 26:0 **SMPx[2:0]**: Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles

001: 2.5 ADC clock cycles

010: 4.5 ADC clock cycles

011: 7.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 61.5 ADC clock cycles

110: 181.5 ADC clock cycles

111: 601.5 ADC clock cycles

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

15.5.7 ADC watchdog threshold register 1 (ADCx_TR1, x=1..2)

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]											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT1[11:0]											
				rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1.

Refer to [Section 15.3.28: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx\)](#)

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

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

Bits 11:0 **LT1[11:0]**: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 1.

Refer to [Section 15.3.28: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx\)](#)

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

15.5.8 ADC watchdog threshold register 2 (ADCx_TR2, x = 1..2)

Address offset: 0x24

Reset value: 0x00FF 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.	HT2[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.	LT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **HT2[7:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 2.

Refer to [Section 15.3.28: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx\)](#)

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

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

Bits 7:0 **LT2[7:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 2.

Refer to [Section 15.3.28: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx\)](#)

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

15.5.9 ADC watchdog threshold register 3 (ADCx_TR3, x=1..2)

Address offset: 0x28

Reset value: 0x00FF 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.	HT3[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.	LT3[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **HT3[7:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to [Section 15.3.28: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx\)](#)

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

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

Bits 7:0 **LT3[7:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3.

This watchdog compares the 8-bit of LT3 with the 8 MSB of the converted data.

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

15.5.10 ADC regular sequence register 1 (ADCx_SQR1, x=1..2)

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.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			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
SQ2[3:0]			Res.	SQ1[4:0]					Res.	Res.	L[3:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]**: 4th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 4th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]**: 3rd conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 3rd in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]**: 2nd conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 2nd in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]**: 1st conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 1st in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]**: Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

0000: 1 conversion

0001: 2 conversions

...

1111: 16 conversions

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

15.5.11 ADC regular sequence register 2 (ADCx_SQR2, x=1..2)

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.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4:0]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
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 28:24 **SQ9[4:0]**: 9th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 9th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]**: 8th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 8th in the regular conversion sequence

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]**: 7th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 7th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]**: 6th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 6th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]**: 5th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 5th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

15.5.12 ADC regular sequence register 3 (ADCx_SQR3, x=1..2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			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
SQ12[3:0]			Res.	SQ11[4:0]					Res.	SQ10[4:0]					
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 28:24 **SQ14[4:0]**: 14th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 14th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]**: 13th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 13th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]**: 12th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 12th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]**: 11th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 11th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]**: 10th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 10th in the regular conversion sequence.

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

15.5.13 ADC regular sequence register 4 (ADCx_SQR4, x=1..2)

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.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

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

Bits 10:6 **SQ16[4:0]**: 16th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 16th in the regular conversion sequence.

*Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).**Note: Analog input channel 0 is not mapped: value "00000" should not be used*

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]**: 15th conversion in regular sequence

These bits are written by software with the channel number (1..18) assigned as the 15th in the regular conversion sequence.

*Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).**Note: Analog input channel 0 is not mapped: value "00000" should not be used*

15.5.14 ADC regular Data Register (ADCx_DR, x=1..2)

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
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **RDATA[15:0]**: Regular Data converted

These bits are read-only. They contain the conversion result from the last converted regular channel.
The data are left- or right-aligned as described in [Section 15.3.26: Data management](#).

15.5.15 ADC injected sequence register (ADCx_JSQR, x=1..2)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:2]		
	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
JSQ2[1:0]		Res.	JSQ1[4:0]				JEXTEN[1:0]		JEXTSEL[3:0]			JL[1:0]			
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 **JSQ4[4:0]**: 4th conversion in the injected sequence

These bits are written by software with the channel number (1..18) assigned as the 4th in the injected conversion sequence.

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **JSQ3[4:0]**: 3rd conversion in the injected sequence

These bits are written by software with the channel number (1..18) assigned as the 3rd in the injected conversion sequence.

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 19 Reserved, must be kept at reset value.

Bits 18:14 **JSQ2[4:0]**: 2nd conversion in the injected sequence

These bits are written by software with the channel number (1..18) assigned as the 2nd in the injected conversion sequence.

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bit 13 Reserved, must be kept at reset value.

Bits 12:8 **JSQ1[4:0]**: 1st conversion in the injected sequence

These bits are written by software with the channel number (1..18) assigned as the 1st in the injected conversion sequence.

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bits 7:6 **JEXTEN[1:0]**: External Trigger Enable and Polarity Selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Note: If JQM=1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to [Section 15.3.21: Queue of context for injected conversions](#))

Bits 5:2 **JEXTSEL[3:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

0000: Event 0

0001: Event 1

0010: Event 2

0011: Event 3

0100: Event 4

0101: Event 5

0110: Event 6

0111: Event 7

...

1111: Event 15

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

Bits 1:0 **JL[1:0]**: Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: Software is allowed to write these bits at any time, once the ADC is enabled (ADEN=1).

15.5.16 ADC offset register (ADCx_OFRy, x=1..2) (y=1..4)

Address offset: 0x60, 0x64, 0x68, 0x6C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSETy_EN	OFFSETy_CH[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	OFFSETy[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

Bit 31 OFFSETy_EN: Offset y Enable

This bit is written by software to enable or disable the offset programmed into bits OFFSETy[11:0].

Note: Software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 30:26 OFFSETy_CH[4:0]: Channel selection for the Data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSETy[11:0] will apply.

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Note: Analog input channel 0 is not mapped: value "00000" should not be used

Bits 25:12 Reserved, must be kept at reset value.**Bits 11:0 OFFSETy[11:0]:** Data offset y for the channel programmed into bits OFFSETy_CH[4:0]

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the bits OFFSETy_CH[4:0]. The conversion result can be read from in the ADCx_DR (regular conversion) or from in the ADCx_JDRyi registers (injected conversion).

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Note: If several offset (OFFSETy) point to the same channel, only the offset with the lowest x value is considered for the subtraction.

Ex: if OFFSET1_CH[4:0]=4 and OFFSET2_CH[4:0]=4, this is OFFSET1[11:0] which is subtracted when converting channel 4.

15.5.17 ADC injected data register (ADCx_JDRy, x=1..2, y= 1..4)

Address offset: 0x80 - 0x8C

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
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **JDATA[15:0]**: Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in [Section 15.3.26: Data management](#).

15.5.18 ADC Analog Watchdog 2 Configuration Register (ADCx_AWD2CR, x=1..2)

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.	AWD2CH[18:16]		
													r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2CH[15:1]															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	r/w	r/w	r/w	

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

Bits 18:1 **AWD2CH[18:1]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWD2CH[i] = 0: ADC analog input channel-i is not monitored by AWD2

AWD2CH[i] = 1: ADC analog input channel-i is monitored by AWD2

When AWD2CH[18:1] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWD2CH must be also selected into the SQRi or JSQRi registers.

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 0 Reserved, must be kept at reset value.

15.5.19 ADC Analog Watchdog 3 Configuration Register (ADCx_AWD3CR, x=1..2)

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.	AWD3CH[18:16]		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3CH[15:1]															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	r/w	r/w	r/w	

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

Bits 18:1 **AWD3CH[18:1]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWD3CH[i] = 0: ADC analog input channel-i is not monitored by AWD3

AWD3CH[i] = 1: ADC analog input channel-i is monitored by AWD3

When AWD3CH[18:1] = 000..0, the analog Watchdog 3 is disabled

Note: The channels selected by AWD3CH must be also selected into the SQRi or JSQRi registers.

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 0 Reserved, must be kept at reset value.

15.5.20 ADC Differential Mode Selection Register (ADCx_DIFSEL, x=1..2)

Address offset: 0xB0

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.	DIFSEL[18:16]		
													r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIFSEL[15:1]															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	r/w	r/w	r/w	

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

Bits 18:16 **DIFSEL[18:16]**: Differential mode for channels 18 to 16.

These bits are read only. These channels are forced to single-ended input mode (either connected to a single-ended I/O port or to an internal channel).

Bits 15:1 **DIFSEL[15:1]**: Differential mode for channels 15 to 1

These bits are set and cleared by software. They allow to select if a channel is configured as single ended or differential mode.

DIFSEL[i] = 0: ADC analog input channel-i is configured in single ended mode

DIFSEL[i] = 1: ADC analog input channel-i is configured in differential mode

Note: Software is allowed to write these bits only when the ADC is disabled (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Note: It is mandatory to keep cleared ADC1_DIFSEL[15] (connected to an internal single ended channel)

Bit 0 Reserved, must be kept at reset value.

15.5.21 ADC Calibration Factors (ADCx_CALFACT, x=1..2)

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.	CALFACT_D[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.	CALFACT_S[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **CALFACT_D[6:0]**: Calibration Factors in differential mode

These bits are written by hardware or by software.

Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it will then be applied once a new differential calibration is launched.

Note: Software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

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

Bits 6:0 **CALFACT_S[6:0]**: Calibration Factors In Single-Ended mode

These bits are written by hardware or by software.

Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it will then be applied once a new single-ended calibration is launched.

Note: Software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

15.6 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

- One set of registers is related to ADC1 (master) and ADC2 (slave)

15.6.1 ADC Common status register (ADCx_CSR, x=12)

Address offset: 0x00 (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADCx_SR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
							Slave ADC								
					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.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
							Master ADC								
					r	r	r	r	r	r	r	r	r	r	r

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

- Bit 26 **JQOVF_SLV**: Injected Context Queue Overflow flag of the slave ADC
This bit is a copy of the JQOVF bit in the corresponding ADCx_ISR register.
- Bit 25 **AWD3_SLV**: Analog watchdog 3 flag of the slave ADC
This bit is a copy of the AWD3 bit in the corresponding ADCx_ISR register.
- Bit 24 **AWD2_SLV**: Analog watchdog 2 flag of the slave ADC
This bit is a copy of the AWD2 bit in the corresponding ADCx_ISR register.
- Bit 23 **AWD1_SLV**: Analog watchdog 1 flag of the slave ADC
This bit is a copy of the AWD1 bit in the corresponding ADCx_ISR register.
- Bit 22 **JEOS_SLV**: End of injected sequence flag of the slave ADC
This bit is a copy of the JEOS bit in the corresponding ADCx_ISR register.
- Bit 21 **JEOC_SLV**: End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADCx_ISR register.
- Bit 20 **OVR_SLV**: Overrun flag of the slave ADC
This bit is a copy of the OVR bit in the corresponding ADCx_ISR register.
- Bit 19 **EOS_SLV**: End of regular sequence flag of the slave ADC
This bit is a copy of the EOS bit in the corresponding ADCx_ISR register.
- Bit 18 **EOC_SLV**: End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADCx_ISR register.

- Bit 17 **EOSMP_SLV**: End of Sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADCx_ISR register.
- Bit 16 **ADRDY_SLV**: Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADCx_ISR register.
- Bits 15:11 Reserved, must be kept at reset value.
- Bit 10 **JQOVF_MST**: Injected Context Queue Overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADCx_ISR register.
- Bit 9 **AWD3_MST**: Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADCx_ISR register.
- Bit 8 **AWD2_MST**: Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADCx_ISR register.
- Bit 7 **AWD1_MST**: Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADCx_ISR register.
- Bit 6 **JEOS_MST**: End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADCx_ISR register.
- Bit 5 **JEOC_MST**: End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADCx_ISR register.
- Bit 4 **OVR_MST**: Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADCx_ISR register.
- Bit 3 **EOS_MST**: End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADCx_ISR register.
- Bit 2 **EOC_MST**: End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADCx_ISR register.
- Bit 1 **EOSMP_MST**: End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADCx_ISR register.
- Bit 0 **ADRDY_MST**: Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADCx_ISR register.

15.6.2 ADC common control register (ADCx_CCR, x=12)

Address offset: 0x08 (this offset address is relative to the master ADC base address + 0x300)

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.	VBAT EN	TS EN	VREF EN	Res.	Res.	Res.	Res.	CKMODE[1:0]	
							rw	rw	rw					rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MDMA[1:0]		DMA CFG	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
rw	rw	rw		rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bit 24 **VBATEN**: V_{BAT} enable

This bit is set and cleared by software to enable/disable the V_{BAT} channel.

0: V_{BAT} channel disabled

1: V_{BAT} channel enabled

Note: Software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 23 **TSEN**: Temperature sensor enable

This bit is set and cleared by software to enable/disable the temperature sensor channel.

0: Temperature sensor channel disabled

1: Temperature sensor channel enabled

Note: Software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} channel.

0: V_{REFINT} channel disabled

1: V_{REFINT} channel enabled

Note: Software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 21:18 Reserved, must be kept at reset value.

Bits 17:16 **CKMODE[1:0]**: ADC clock mode

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

00: CK_ADCx (x=123) (Asynchronous clock mode), generated at product level (refer to [Section 8: Reset and clock control \(RCC\)](#))

01: HCLK/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set to 1 (HPRE[3:0] = 0xxx in RCC_CFGR register) and if the system clock has a 50% duty cycle.

10: HCLK/2 (Synchronous clock mode)

11: HCLK/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

Note: Software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 15:14 **MDMA[1:0]**: Direct memory access mode for dual ADC mode

This bit-field is set and cleared by software. Refer to the DMA controller section for more details.

00: MDMA mode disabled

01: reserved

10: MDMA mode enabled for 12 and 10-bit resolution

11: MDMA mode enabled for 8 and 6-bit resolution

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 13 **DMACFG**: DMA configuration (for dual ADC mode)

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN=1.

0: DMA One Shot Mode selected

1: DMA Circular Mode selected

For more details, refer to [Section : Managing conversions using the DMA](#)

Note: Software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY**: Delay between 2 sampling phases

Set and cleared by software. These bits are used in dual interleaved modes. Refer to [Table 94](#) for the value of ADC resolution versus DELAY bits values.

Note: Software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

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

Bits 4:0 **DUAL[4:0]**: Dual ADC mode selection

These bits are written by software to select the operating mode.

All the ADCs independent:

00000: Independent mode

00001 to 01001: Dual mode, master and slave ADCs working together

00001: Combined regular simultaneous + injected simultaneous mode

00010: Combined regular simultaneous + alternate trigger mode

00011: Combined Interleaved mode + injected simultaneous mode

00100: Reserved

00101: Injected simultaneous mode only

00110: Regular simultaneous mode only

00111: Interleaved mode only

01001: Alternate trigger mode only

All other combinations are reserved and must not be programmed

Note: Software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Table 94. DELAY bits versus ADC resolution

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * T _{ADC_CLK}	1 * T _{ADC_CLK}	1 * T _{ADC_CLK}	1 * T _{ADC_CLK}
0001	2 * T _{ADC_CLK}	2 * T _{ADC_CLK}	2 * T _{ADC_CLK}	2 * T _{ADC_CLK}
0010	3 * T _{ADC_CLK}	3 * T _{ADC_CLK}	3 * T _{ADC_CLK}	3 * T _{ADC_CLK}
0011	4 * T _{ADC_CLK}	4 * T _{ADC_CLK}	4 * T _{ADC_CLK}	4 * T _{ADC_CLK}
0100	5 * T _{ADC_CLK}	5 * T _{ADC_CLK}	5 * T _{ADC_CLK}	5 * T _{ADC_CLK}
0101	6 * T _{ADC_CLK}	6 * T _{ADC_CLK}	6 * T _{ADC_CLK}	6 * T _{ADC_CLK}
0110	7 * T _{ADC_CLK}	7 * T _{ADC_CLK}	7 * T _{ADC_CLK}	6 * T _{ADC_CLK}
0111	8 * T _{ADC_CLK}	8 * T _{ADC_CLK}	8 * T _{ADC_CLK}	6 * T _{ADC_CLK}
1000	9 * T _{ADC_CLK}	9 * T _{ADC_CLK}	8 * T _{ADC_CLK}	6 * T _{ADC_CLK}
1001	10 * T _{ADC_CLK}	10 * T _{ADC_CLK}	8 * T _{ADC_CLK}	6 * T _{ADC_CLK}
1010	11 * T _{ADC_CLK}	10 * T _{ADC_CLK}	8 * T _{ADC_CLK}	6 * T _{ADC_CLK}
1011	12 * T _{ADC_CLK}	10 * T _{ADC_CLK}	8 * T _{ADC_CLK}	6 * T _{ADC_CLK}
others	12 * T _{ADC_CLK}	10 * T _{ADC_CLK}	8 * T _{ADC_CLK}	6 * T _{ADC_CLK}

15.6.3 ADC common regular data register for dual mode (ADCx_CDR, x=12)

Address offset: 0x0C (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

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

Bits 31:16 **RDATA_SLV[15:0]**: Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to [Section 15.3.29: Dual ADC modes \(STM32F302xB/C only\)](#).

The data alignment is applied as described in [Section : Data register, data alignment and offset \(ADCx_DR, OFFSETy, OFFSETy_CH, ALIGN\)](#)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to [Section 15.3.29: Dual ADC modes \(STM32F302xB/C only\)](#).

The data alignment is applied as described in [Section : Data register, data alignment and offset \(ADCx_DR, OFFSETy, OFFSETy_CH, ALIGN\)](#)

In MDMA=0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

15.6.4 ADC register map

The following table summarizes the ADC registers.

Table 95. ADC global register map

Offset	Register
0x000 - 0x04C	Master ADC1
0x050 - 0x0FC	Reserved
0x100 - 0x14C	Slave ADC2
0x118 - 0x1FC	Reserved
0x200 - 0x24C	Reserved
0x250 - 0x2FC	Reserved
0x300 - 0x308	Master and slave ADCs common registers (ADC12)

Table 96. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC, x=1..2)

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	ADCx_ISR	Res.	Res.																					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x04	ADCx_IER	Res.	Res.																					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x08	ADCx_CR	ADCAL	ADCALDIF	ADVREGEN[1:0]																													
	Reset value	0	0	1	0																												
0x0C	ADCx_CFGR	Res.	AWD1CH[4:0]				AUTO		JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM [2:0]		DISCEN		Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL [3:0]			ALIGN		RES [1:0]	Res.	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	0	0	0	0	0
0x10	Reserved	Res.																															
0x14	ADCx_SMPR1	Res.	Res.	SMP9 [2:0]		SMP8 [2:0]		SMP7 [2:0]		SMP6 [2:0]		SMP5 [2:0]		SMP4 [2:0]		SMP3 [2:0]		SMP2 [2:0]		SMP1 [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	0	0	0	0	0	0	0	0	0	0	0	0
0x18	ADCx_SMPR2	Res.	Res.	Res.	Res.	Res.	SMP18 [2:0]		SMP17 [2:0]		SMP16 [2:0]		SMP15 [2:0]		SMP14 [2:0]		SMP13 [2:0]		SMP12 [2:0]		SMP11 [2:0]		SMP10 [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
0x1C	Reserved	Res.																															
0x20	ADCx_TR1	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							0	0	0	0	0	0	0	0	0	0	0
0x24	ADCx_TR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT2[7:0]							Res.	Res.	Res.	Res.	Res.	LT2[7:0]											
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0
0x28	ADCx_TR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT3[7:0]							Res.	Res.	Res.	Res.	Res.	Res.	LT3[7:0]										
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0
0x2C	Reserved	Res.																															
0x30	ADCx_SQR1	Res.	Res.	Res.	SQ4[4:0]				Res.	SQ3[4:0]				Res.	SQ2[4:0]				Res.	SQ1[4:0]				Res.	Res.	Res.	L[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
0x34	ADCx_SQR2	Res.	Res.	Res.	SQ9[4:0]				Res.	SQ8[4:0]				Res.	SQ7[4:0]				Res.	SQ6[4:0]				Res.	Res.	Res.	SQ5[4:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	ADCx_SQR3	Res.	Res.	Res.	SQ14[4:0]				Res.	SQ13[4:0]				Res.	SQ12[4:0]				Res.	SQ11[4:0]				Res.	Res.	Res.	SQ10[4:0]						
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	ADCx_SQR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]				Res.	SQ15[4:0]						
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x40	ADCx_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	regular RDATA[15:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x44-0x48	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.
0x4C	ADCx_JSQR	Res.	JSQ4[4:0]				Res.	JSQ3[4:0]				Res.	JSQ2[4:0]				Res.	JSQ1[4:0]				JEXTEN[1:0]		JEXTSEL [3:0]			JL[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
0x50-0x5C	Reserved	Res.																															

Table 96. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC, x=1..2) (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		
0x60	ADCx_OFR1	OFFSET1_EN	OFFSET1_CH[4:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET1[11:0]													
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0		
0x64	ADCx_OFR2	OFFSET2_EN	OFFSET2_CH[4:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET2[11:0]													
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0		
0x68	ADCx_OFR3	OFFSET3_EN	OFFSET3_CH[4:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET3[11:0]													
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0		
0x6C	ADCx_OFR4	OFFSET4_EN	OFFSET4_CH[4:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET4[11:0]													
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0		
0x70-0x7C	Reserved	Res.																																	
0x80	ADCx_JDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA1[15:0]																	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x84	ADCx_JDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA2[15:0]																	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x88	ADCx_JDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA3[15:0]																	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x8C	ADCx_JDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA4[15:0]																	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x8C-0x9C	Reserved	Res.																																	
0xA0	ADCx_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[18:1]																		Res.		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xA4	ADCx_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[18:1]																		Res.		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xA8-0xAC	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.		
0xB0	ADCx_DIFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[18:1]																		Res.		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xB4	ADCx_CALFACT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[6:0]													
	Reset value										0	0	0	0	0	0	0											0	0	0	0	0	0		

Table 97. ADC register map and reset values (master and slave ADC common registers) offset =0x300, x=1 or 34)

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	ADCx_CSR	Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
		slave ADC2																master ADC1															
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x04	Reserved	Res.																															
0x08	ADCx_CCR	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	Res.	Res.	Res.	Res.	CKMODE[1:0]	MDMA[1:0]	DMACFG	Res.	DELAY[3:0]			Res.	Res.	Res.	DUAL[4:0]								
	Reset value							0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0
0x0C	ADCx_CDR	RDATA_SLV[15:0]																RDATA_MST[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

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

16 Digital-to-analog converter (DAC1)

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 could be left- or right-aligned. An input reference voltage, V_{REF+} (shared with ADC), is available. The output can optionally be buffered for higher current drive.

16.2 DAC1 main features

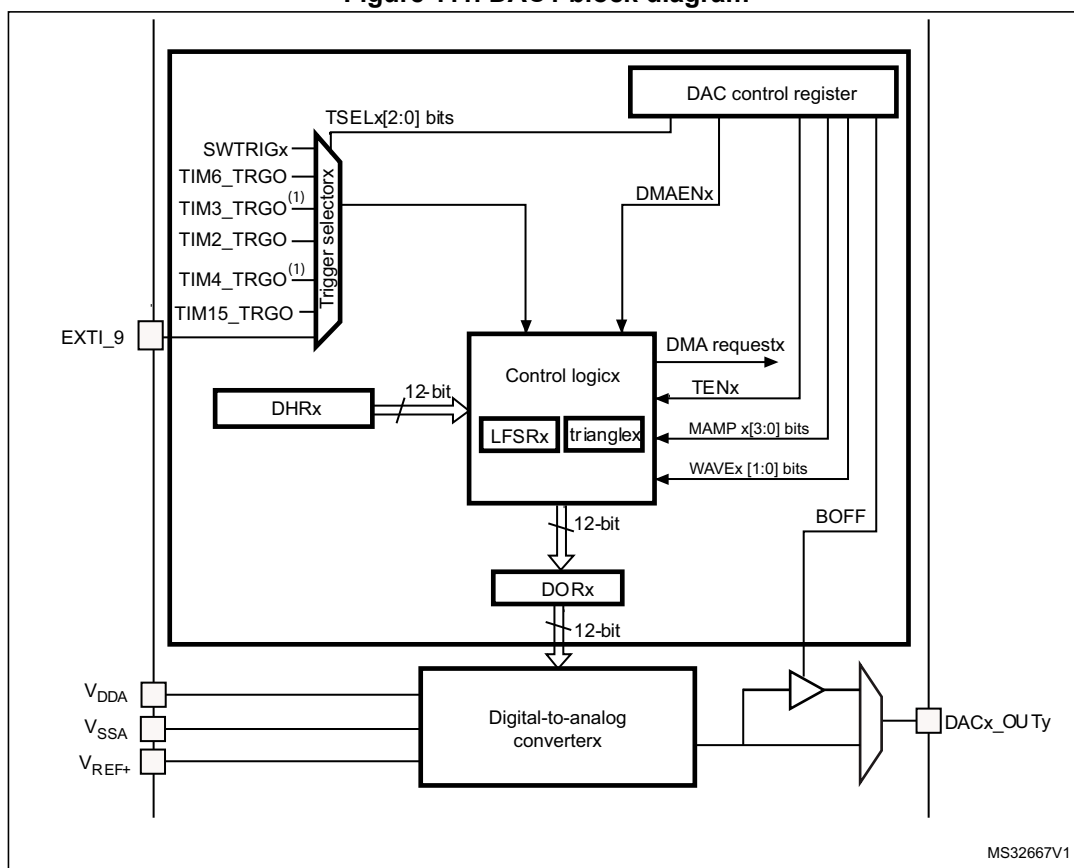
- DAC1 integrates one 12-bit DAC channel DAC1_OUT1

The DAC main features are the following:

- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave generation
- Triangular-wave generation
- DMA capability for each channel
- DMA underrun error detection
- External triggers for conversion
- Programmable internal buffer
- Input voltage reference, V_{DDA}

[Figure 111](#) shows the block diagram of the DAC1 channel and [Table 98](#) gives the pin description.

Figure 111. DAC1 block diagram



1. TIM3_TRGO and TIM4_TRGO are only available on STM32F302xB/C/D/E devices.

Table 98. DAC1 pins

Name	Signal type	Remarks
V _{DDA}	Input, analog supply	Analog power supply
V _{SSA}	Input, analog supply ground	Ground for analog power supply
DAC1_OUT1	Analog output signal	DAC1 channel 1 analog output

Note: Once the DAC1 channel 1 is enabled, the corresponding GPIO pin (PA4) is automatically connected to the analog converter output (DACx_OUTy). In order to avoid parasitic consumption, the PA4, pin should first be configured to analog (AIN).

16.3 Single mode functional description

16.3.1 DAC channel enable

The DAC channel can be powered on by setting the EN1 bit in the DAC_CR register. The DAC channel is then enabled after a startup time t_{WAKEUP} .

Note: The EN1 bit enables the analog DAC Channel macrocell only. The DAC Channel digital interface is enabled even if the EN1 bit is reset.

16.3.2 DAC output buffer enable

The DAC integrates one output buffer that can be used to reduce the output impedance, and to drive external loads directly without having to add an external operational amplifier. The DAC channel output buffer can be enabled and disabled through the BOFF1 bit in the DAC_CR register.

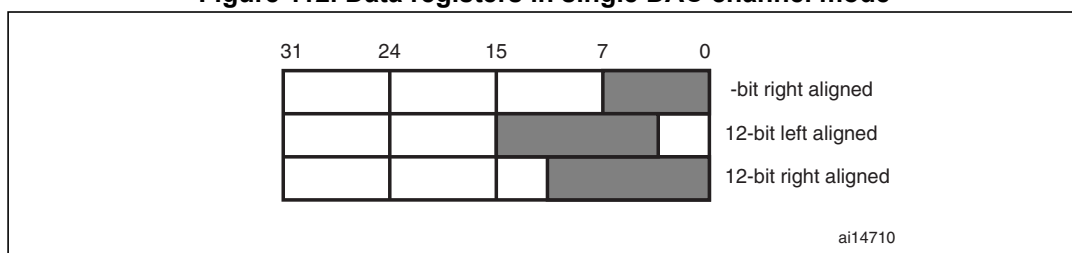
16.3.3 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- There are three possibilities:
 - 8-bit right alignment: the software has to load data into the DAC_DHR8Rx [7:0] bits (stored into the DHRx[11:4] bits)
 - 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)
 - 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DHRx (data holding registerx, which are internal non-memory-mapped registers). The DHRx register is then loaded into the DORx register either automatically, by software trigger or by an external event trigger.

Figure 112. Data registers in single DAC channel mode



16.3.4 DAC channel conversion

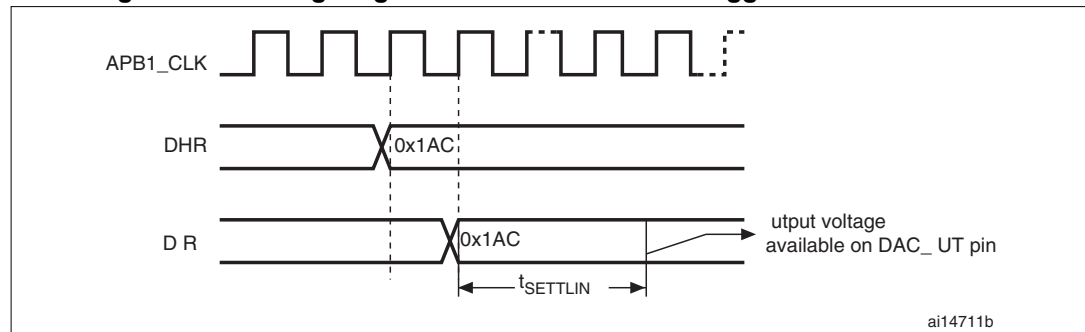
The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one APB1 clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR

register is set) and a trigger occurs, the transfer is performed three PCLK1 clock cycles later.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time t_{SETTLING} that depends on the power supply voltage and the analog output load.

Figure 113. Timing diagram for conversion with trigger disabled $TEN = 0$



Independent trigger with single LFSR generation

To configure the DAC in this conversion mode (see [Section 16.4: Noise generation](#)), the following sequence is required:

1. Set the DAC channel trigger enable bit $TENx$.
2. Configure the trigger source by setting $TSELx[2:0]$ bits.
3. Configure the DAC channel $WAVEx[1:0]$ bits as "01" and the same LFSR mask value in the $MAMPx[3:0]$ bits.
4. Load the DAC channel data into the desired DAC_DHRx register (DHR12RD, DHR12LD or DHR8RD).

When a DAC channelx trigger arrives, the LFSRx counter, with the same mask, is added to the DHRx register and the sum is transferred into DAC_DORx (three APB clock cycles later). Then the LFSRx counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode (see [Section 16.5: Triangle-wave generation](#)), the following sequence is required:

1. Set the DAC channelx trigger enable $TENx$ bits.
2. Configure the trigger source by setting $TSELx[2:0]$ bits.
3. Configure the DAC channelx $WAVEx[1:0]$ bits as "1x" and the same maximum amplitude value in the $MAMPx[3:0]$ bits.
4. Load the DAC channelx data into the desired DAC_DHRx register. (DHR12RD, DHR12LD or DHR8RD).

When a DAC channelx trigger arrives, the DAC channelx triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC_DORx (three APB clock cycles later). The DAC channelx triangle counter is then updated.

16.3.5 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+} (or V_{DDA} depending on the package).

The analog output voltages on each DAC channel pin are determined by the following equation:

$$DAC_{output} = V_{REF+} \times \frac{DOR}{4095}$$

Note: (V_{REF+} is replaced by V_{DDA} on the 64-, 49-, 48- and 32-pin packages)

16.3.6 DAC trigger selection

If the TENx control bit is set, conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[2:0] control bits determine which out of 8 possible events will trigger conversion as shown in [Table 99](#).

Table 99. External triggers (DAC1)

Source	Type	TSEL[2:0]
TIM6_TRGO event	Internal signal from on-chip timers	000
TIM3_TRGO event ⁽¹⁾		001
Reserved		010
TIM15_TRGO event		011
TIM2_TRGO event		100
TIM4_TRGO event		101
EXTI line9	External pin	110
SWTRIG	Software control bit	111

1. To select TIM3_TRGO event as DAC1 trigger source, the DAC_TRIG_RMP bit must be set in SYSCFG_CFGR1 register.

Each time a DAC interface detects a rising edge on the selected timer TRGO output, or on the selected external interrupt line 9, the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three APB1 cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[2:0] bit cannot be changed when the ENx bit is set. When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one APB1 clock cycle.

$$V_{REF+}^{DAC_{output}} = V_{REF+} \times \frac{DOR}{4095}$$

Table 100. External triggers (DAC1)

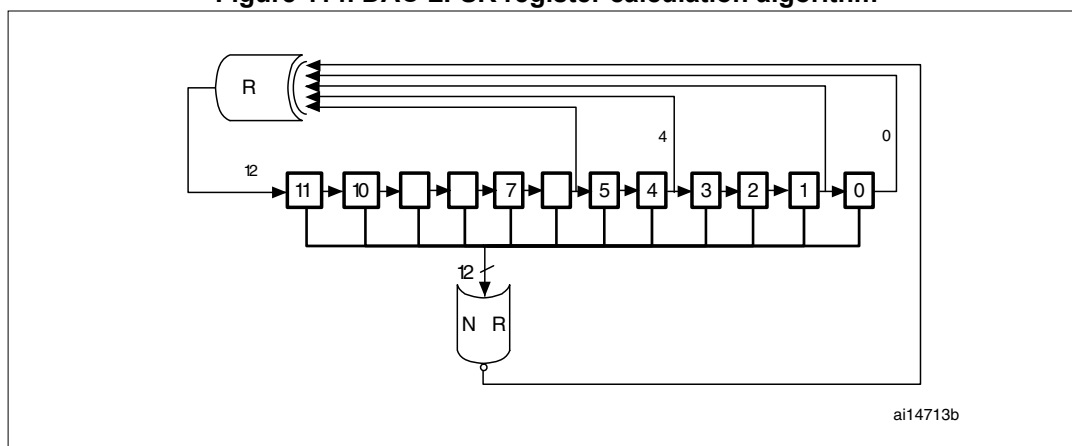
Source	Type	TSEL[2:0]
TIM6_TRGO event	Internal signal from on-chip timers	000
TIM3_TRGO event ⁽¹⁾		001
Reserved		010
TIM15_TRGO event		011
TIM2_TRGO event		100
TIM4_TRGO event		101
EXTI line9	External pin	110
SWTRIG	Software control bit	111

1. To select TIM3_TRGO event as DAC1 trigger source, the DAC_TRIG_RMP bit must be set in SYSCFG_CFGR1 register.

16.4 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to "01". The preloaded value in LFSR is 0xAAA. This register is updated three APB clock cycles after each trigger event, following a specific calculation algorithm.

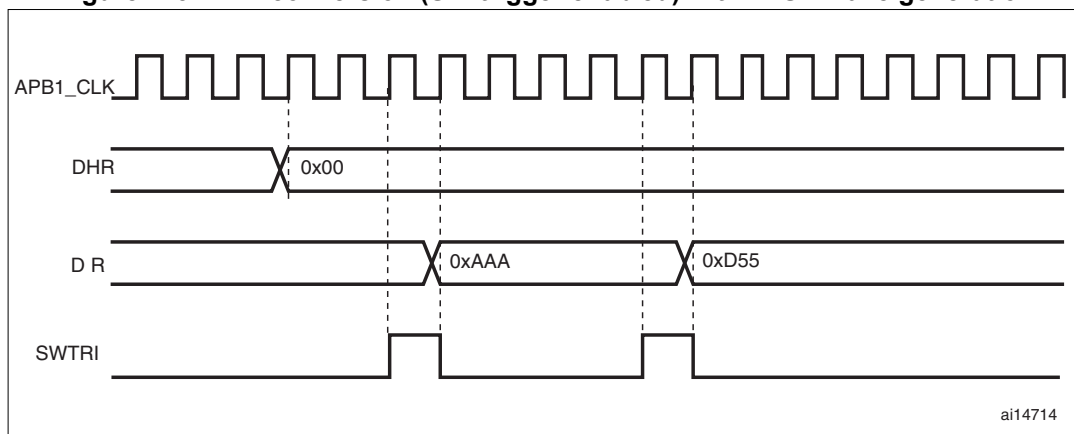
Figure 114. DAC LFSR register calculation algorithm



The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then stored into the DAC_DORx register.

If LFSR is 0x0000, a '1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 115. 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.

16.5 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting *WAVEx*[1:0] to “10”. The amplitude is configured through the *MAMPx*[3:0] bits in the *DAC_CR* register. An internal triangle counter is incremented three APB clock cycles after each trigger event. The value of this counter is then added to the *DAC_DHRx* register without overflow and the sum is stored into the *DAC_DORx* register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the *MAMPx*[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the *WAVEx*[1:0] bits.

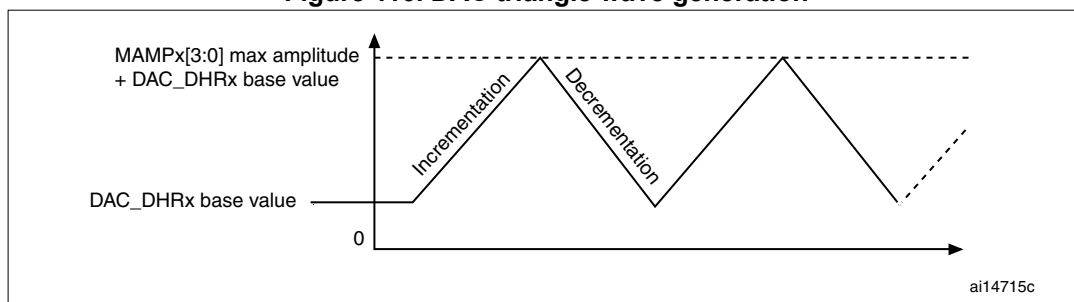
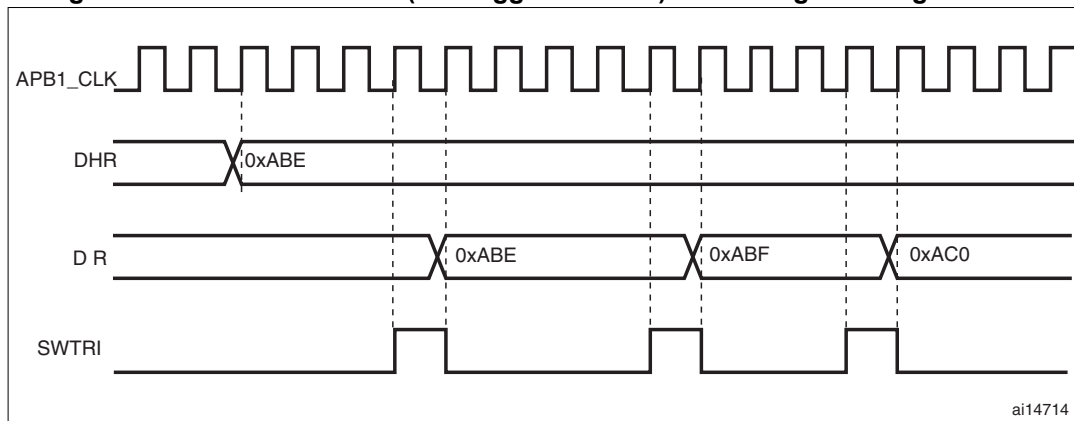
Figure 116. DAC triangle wave generation

Figure 117. DAC conversion (SW trigger enabled) with triangle wave generation

Note: The DAC trigger must be enabled for triangle generation by setting the *TENx* bit in the *DAC_CR* register.

The *MAMPx[3:0]* bits must be configured before enabling the DAC, otherwise they cannot be changed.

16.6 DMA request

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

A DAC DMA request is generated when an external trigger (but not a software trigger) occurs while the *DMAENx* bit is set. The value of the *DAC_DHRx* register is then transferred to the *DAC_DORx* register.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag *DMAUDRx* in the *DAC_SR* register is set, reporting the error condition. DMA data transfers are then disabled and no further DMA request is treated. The DAC channelx continues to convert old data.

The software should clear the *DMAUDRx* flag by writing "1", clear the *DMAEN* bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software should modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channel, an interrupt is also generated if the corresponding *DMAUDRIEx* bit in the *DAC_CR* register is enabled.

16.7 DAC registers

Refer to [Section 2.1 on page 42](#) 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.	Res.	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.	DMAU DRIE1	DMA EN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

00: Wave generation disabled

01: Noise wave generation enabled

1x: Triangle wave generation enabled

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:3 **TSEL1[2:0]**: DAC channel1 trigger selection

These bits select the external event used to trigger DAC channel1.

000: Timer 6 TRGO event

001: Timer 3 TRGO event depending on the value of DAC_TRIG_RMP bit in SYSCFG_CFGR1 register (STM32F302x6/8 only)

010: Reserved

011: Timer 15 TRGO event

100: Timer 2 TRGO event

101: Timer 4 TRGO event (STM32F302x6/8 only)

110: EXTI line9

111: Software trigger

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 2 **TEN1**: DAC channel1 trigger enable

This bit is set and cleared by software to enable/disable DAC channel1 trigger.

0: DAC channel1 trigger disabled and data written into the DAC_DHRx register are transferred one APB1 clock cycle later to the DAC_DOR1 register

1: DAC channel1 trigger enabled and data from the DAC_DHRx register are transferred three APB1 clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DOR1 register takes only one APB1 clock cycle.

Bit 1 **BOFF1**: DAC channel1 output buffer disable

This bit is set and cleared by software to enable/disable DAC channel1 output buffer.

0: DAC channel1 output buffer enabled

1: DAC channel1 output buffer disabled

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.

0: DAC channel1 disabled

1: DAC channel1 enabled

16.7.2 DAC software trigger register (DAC_SWTRIGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	SWTRIG1
															w

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

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set and cleared by software to enable/disable the software trigger.

0: Software trigger disabled

1: Software trigger enabled

Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

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.	DACC1DHR[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 which specifies 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
DACC1DHR[11:0]												v	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 which specifies 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 which specifies 8-bit data for DAC channel1.

16.7.6 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.	Res.	Res.	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.

16.7.7 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rc_w1													

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

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bits 12:0 Reserved, must be kept at reset value.

16.7.8 DAC register map

Table 101 summarizes the DAC registers.

Table 101. DAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	DAC_CR	Res.	Res.	Res.	Res.	Res.				Res.	Res.				Res.	Res.	Res.	Res.	Res.	DMAUDRIE1	DMAEN1	MAMP1[3:0]			WAVE1[1:0]		TSEL1[2:0]		TEN1	BOFF1	EN1			
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	DAC_SWTRIGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIGR1	
	Reset value																																0	
0x08	DAC_DHR12R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]													
	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.	DACC1DHR[11:0]													Res.	Res.
	Reset value																		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.	DACC1DHR[7:0]								
	Reset value																										0	0	0	0	0	0	0	0
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DOR[11:0]													
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0	
0x34	DAC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																			0														

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

17 Comparator (COMP)

17.1 Introduction

STM32F302xB/C/D/E and STM32F302x6/8 embed four general purpose comparators that can be used either as standalone devices (all terminal are available on I/Os) or combined with the timers. STM32F302x6/8 embed three comparators, COMP2, COMP4 and COMP6.

STM32F302xx embed four general purpose comparators COMP1, COMP2, COMP4 and COMP6 that can be used either as standalone devices (all terminal are available on I/Os) or combined with the timers.

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 the DAC and a PWM output from a timer.

17.2 COMP main features

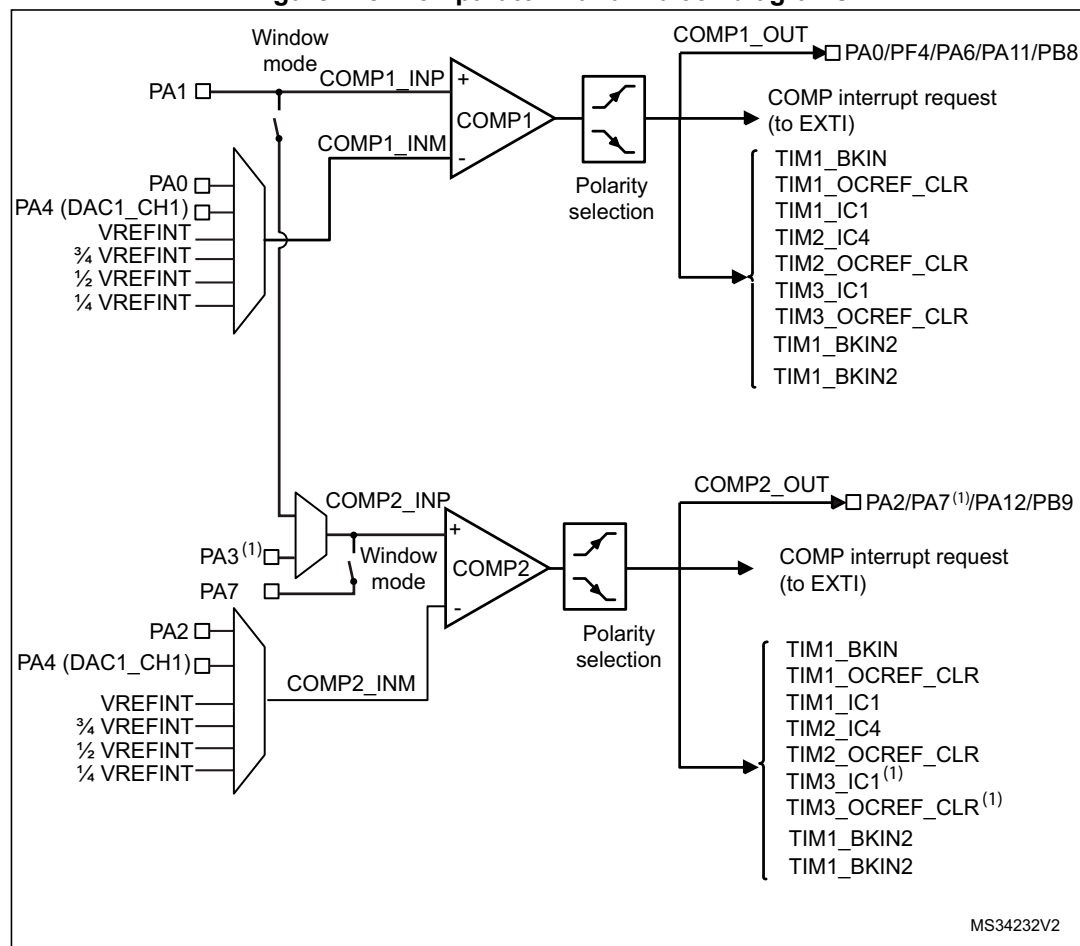
- Rail-to-rail comparators
- Each comparator has positive and configurable negative inputs used for flexible voltage selection:
 - Multiplexed I/O pins
 - DAC1 channel 1
 - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- Programmable speed / consumption (only on STM32F302xBxC)
- Programmable hysteresis (only on STM32F302xBxC)
- The outputs can be redirected to an I/O or to multiple timer inputs for triggering:
 - Capture events
 - OCREF_CLR events (for cycle-by-cycle current control)
 - Break events for fast PWM shutdowns
- COMP1 and COMP2 comparators can be combined in a window comparator. This applies to STM32F302xB/C/D/E devices only.
- Comparator outputs with blanking source
- Each comparator has interrupt generation capability with wake-up from Sleep and Stop modes (through the EXTI controller)

17.3 COMP functional description

17.3.1 COMP block diagram

The block diagram of the comparators is shown in [Figure 118: Comparator 1 and 2 block diagrams](#).

Figure 118. Comparator 1 and 2 block diagrams



1. Available only in STM32F302xB/C.

17.3.2 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 table below summarizes the I/Os that can be used as comparators inputs and outputs.

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 102. STM32F302xB/C/D/E comparator input/outputs summary

	Comparator input/outputs			
	COMP1	COMP2	COMP4	COMP6
Comparator inverting Input: connection to internal signals	DAC1_CH1 Vrefint $\frac{3}{4}$ Vrefint $\frac{1}{2}$ Vrefint $\frac{1}{4}$ Vrefint			
Comparator Inputs connected to I/Os (+: non inverting input; -: inverting input)	+: PA1 -: PA0	+: PA3 ⁽¹⁾ +: PA7 -: PA2	+: PB0 +: PE7 ⁽¹⁾ -: PB2 -: PE8	+: PB11 +: PD11 ⁽¹⁾ -: PB15 -: PD10
Comparator outputs (motor control protection)	T1BKIN T1BKIN2 T1BKIN2			
Outputs on I/Os	PA0 PF4 PA6 PA11 PB8	PA2 PA7 ⁽¹⁾ PA12 PB9	PB1	PA10 PC6
Outputs to internal signals	TIM1_OCrefClear TIM1_IC1 TIM2_IC4 TIM2_OCrefClear TIM3_IC1 TIM3_OCrefClear		TIM3_IC3 TIM3_OCrefClear TIM4_IC2 TIM15_OCrefClear TIM15_IC2	TIM2_IC2 TIM2_OCrefClear TIM16_OCrefClear TIM16_IC1 TIM4_IC4

1. Only on STM32F302xB/C devices.

Table 103. STM32F302x6/8 comparator input/outputs summary

	Comparator input/outputs		
	COMP2	COMP4	COMP6
Comparator inverting Input: connection to internal signals	DAC1_CH1 Vrefint $\frac{3}{4}$ Vrefint $\frac{1}{2}$ Vrefint $\frac{1}{4}$ Vrefint		
Comparator Inputs connected to I/Os (+: non inverting input; -: inverting input)	+: PA7 -: PA2	+: PB0 -: PB2	+: PB11 -: PB15
Comparator outputs (motor control protection)	T1BKIN T1BKIN2		
Outputs on I/Os	PA2 PA12 PB12	PB1	PA10 PC6
Outputs to internal signals	TIM1_OCREF_CLR TIM1_IC1 TIM2_IC4 TIM2_OCREF_CLR	TIM15_OCREF_CLR TIM15_IC2	TIM2_IC2 TIM2_OCREF_CLR TIM16_OCREF_CLR TIM16_IC1

17.3.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the PCLK2 (APB2 clock).

There is no clock enable control bit provided in the RCC controller. To use a clock source for the comparator, the SYSCFG clock enable control bit must be set in the RCC controller.

Note: ***Important:** The polarity selection logic and the output redirection to the port works independently from the PCLK2 clock. This allows the comparator to work even in Stop mode.*

17.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary to insure that the comparator programming cannot be altered in case of spurious register access or program counter corruption.

For this purpose, the comparator control and status registers can be write-protected (read-only).

Once the programming is completed, using bits 30:0 of COMPx_CSR, the COMPxLOCK bit can be set to 1. This causes the whole COMPx_CSR register to become read-only, including the COMPxLOCK bit.

The write protection can only be reset by a MCU reset.

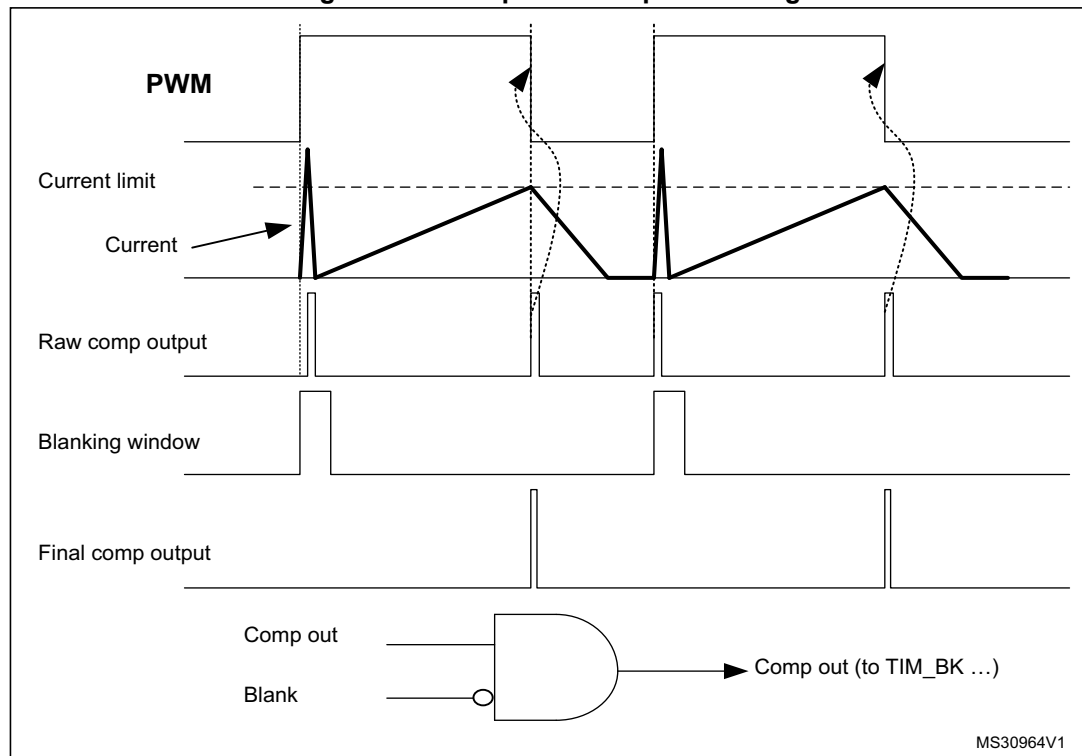
17.3.5 Hysteresis (STM32F302xBxC only)

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.

17.3.6 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 119. Comparator output blanking



MS30964V1

17.3.7 Power mode (STM32F302xB/C only)

The comparator power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application. The bits COMPxMODE[1:0] in COMPx_CSR registers can be programmed as follows:

- 00: High speed / full power
- 01: Medium speed / medium power
- 10: Low speed / low-power
- 11: Very-low speed / ultra-low-power

17.4 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.

17.5 COMP registers

17.5.1 COMP1 control and status register (COMP1_CSR)

Note: This register is available in STM32F302xB/C/D/E only

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP1 LOCK	COMP1 OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP1 BLANKING			COMP1HYST [1:0] ⁽¹⁾	
rw	r										rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP1 POL	Res.	COMP1OUTSEL				Res.	Res.	Res.	COMP1INMSEL[2:0]			COMP1MODE [1:0] ⁽¹⁾		COMP1_INP_DAC	COMP1EN
rw		rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw

1. Only in STM32F302xB/C.

Bit 31 **COMP1LOCK**: Comparator 1 lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

It allows to have COMP1_CSR register as read-only.

0: COMP1_CSR is read-write.

1: COMP1_CSR is read-only.

Bit 30 **COMP1OUT**: Comparator 1 output

This read-only bit is a copy of comparator 1 output state.

0: Output is low (non-inverting input below inverting input).

1: Output is high (non-inverting input above inverting input).

Bits 29:21 Reserved, must be kept at reset value.

Bits 20:18 **COMP1_BLANKING**: Comparator 1 blanking source

These bits select which Timer output controls the comparator 1 output blanking.

000: No blanking

001: TIM1 OC5 selected as blanking source

010: TIM2 OC3 selected as blanking source

011: TIM3 OC3 selected as blanking source

Other configurations: reserved

Bits 17:16 **COMP1HYST[1:0]** Comparator 1 hysteresis

These bits control the hysteresis level.

00: No hysteresis

01: Low hysteresis

10: Medium hysteresis

11: High hysteresis

Please refer to the electrical characteristics for the hysteresis values.

Bit 15 **COMP1POL**: Comparator 1 output polarity

This bit is used to invert the comparator 1 output.

0: Output is not inverted

1: Output is inverted

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **COMP1OUTSEL[3:0]**: Comparator 1 output selection

These bits select which Timer input must be connected with the comparator1 output.

0000: No selection
 0001: (BRK_ACTH) Timer 1 break input
 0010: (BRK2) Timer 1 break input 2
 0101: Timer 1 break input 2
 0110: Timer 1 OCrefclear input
 0111: Timer 1 input capture 1
 1000: Timer 2 input capture 4
 1001: Timer 2 OCrefclear input
 1010: Timer 3 input capture 1
 1011: Timer 3 OCrefclear input
 Remaining combinations: reserved.

Bits 9:7 Reserved, must be kept at reset value.

Bits 6:4 **COMP1INMSEL[2:0]**: Comparator 1 inverting input selection

These bits allows to select the source connected to the inverting input of the comparator 1.

000: 1/4 of Vrefint
 001: 1/2 of Vrefint
 010: 3/4 of Vrefint
 011: Vrefint
 100: PA4 or DAC1 output if enabled
 101: PA5
 110: PA0
 111: Reserved

Bits 3:2 **COMP1MODE[1:0]**: Comparator 1 mode

These bits control the operating mode of the comparator1 and allows to adjust the speed/consumption.

00: High speed
 01: Medium speed
 10: Low-power
 11: Ultra-low-power

Bit 1 **COMP1_INP_DAC**: Comparator 1 non inverting input connection to DAC output.

This bit closes a switch between comparator 1 non-inverting input (PA0) and DAC out I/O (PA4).

0: Switch open
 1: Switch closed

Note: This switch is solely intended to redirect signals onto high impedance input, such as COMP1 non-inverting input (highly resistive switch).

Bit 0 **COMP1EN**: Comparator 1 enable

This bit switches COMP1 ON/OFF.

0: Comparator 1 disabled
 1: Comparator 1 enabled

17.5.2 COMP2 control and status register (COMP2_CSR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP2LOCK	COMP2OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP2_BLANKING[2:0]			COMP2HYST[1:0] ⁽¹⁾	
rw	r										rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP2POL	Res.	COMP2OUTSEL[3:0]				COMP2WINMODE ⁽²⁾	Res.	COMP2INPSEL ⁽¹⁾	COMP2INMSEL[2:0]			COMP2MODE[1:0] ⁽¹⁾		COMP2_INP_DAC ⁽³⁾	COMP2EN
rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

1. Only in STM32F302xB/C devices.
2. Only in STM32F302xB/C/D/E devices.
3. Only in STM32F302x6/8 devices.

Bit 31 COMP2LOCK: Comparator 2 lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.
It allows to have COMP2_CSR register as read-only.
0: COMP2_CSR is read-write.
1: COMP2_CSR is read-only.

Bit 30 COMP2OUT: Comparator 2 output

This read-only bit is a copy of comparator 1 output state.
0: Output is low (non-inverting input below inverting input).
1: Output is high (non-inverting input above inverting input).

Bits 29:21 Reserved, must be kept at reset value.

Bits 20:18 COMP2_BLANKING[2:0]: Comparator 2 output blanking source

These bits select which Timer output controls the comparator 1 output blanking.
000: No blanking
001: TIM1 OC5 selected as blanking source
010: TIM2 OC3 selected as blanking source
011: TIM3 OC3 selected as blanking source
Other configurations: reserved

Bits 17:16 COMP2HYST[1:0]: Comparator 2 Hysteresis

On the STM32F302xB/C, these bits control the hysteresis level.
00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Please refer to the electrical characteristics for the hysteresis values.

On the STM32F302x6/8, these bits are reserved and must be kept at reset value.

Bit 15 COMP2POL: Comparator 2 output polarity

This bit is used to invert the comparator 2 output.
0: Output is not inverted
1: Output is inverted

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **COMP2OUTSEL[3:0]**: Comparator 2 output selection

These bits select which Timer input must be connected with the comparator2 output.

0000: No selection

0001: (BRK_ACTH) Timer 1 break input

0010: (BRK2) Timer 1 break input 2

0011: Reserved

0100: Reserved

0101: Timer 1 break input2

0110: Timer 1 OCREF_CLR input

0111: Timer 1 input capture 1

1000: Timer 2 input capture 4

1001: Timer 2 OCREF_CLR input

1010: Timer 3 input capture 1

1011: Timer 3 OCrefclear input

Remaining combinations: reserved.

Bit 9 **COMP2WINMODE**: Comparator 2 window mode (Only in STM32F302xB/C/D/E devices)

This bit selects the window mode: Both non inverting inputs of comparators share the non inverting input of Comparator 1 (PA1).

0: Comparators 1 and 2 can not be used in window mode.

1: Comparators 1 and 2 can be used in window mode.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **COMP2INPSEL**: Comparator 2 non inverting input selection (Only in STM32F302xB/C devices)

0: PA7 is selected.

1: PA3 is selected.

Note: On STM32F302x6/x8 and STM32F302xD/E, this bit is reserved. COMP2_VINP is available on PA3 whatever value is written in bit 7.

Bits 6:4 **COMP2INMSEL[2:0]**: Comparator 2 inverting input selection

These bits allows to select the source connected to the inverting input of the comparator 2.

0000: 1/4 of Vrefint

0001: 1/2 of Vrefint

0010: 3/4 of Vrefint

0011: Vrefint

0100: PA4 or DAC1_CH1 output if enabled

0110: PA2

Remaining combinations: reserved.

Bits 3:2 **COMP2MODE[1:0]**: Comparator 2 mode (Only in STM32F302xB/C devices)

These bits control the operating mode of the comparator2 and allows to adjust the speed/consumption.

00: High speed

01: Medium speed

10: Low-power

11: Ultra-low-power

Bit 1 **COMP2_INP_DAC**: Comparator 2 non inverting input connection to DAC output. (STM32F302x6/8 devices only)

This bit closes a switch between comparator 2 non-inverting input and DAC out I/O.

0: Switch open

1: Switch closed

This switch is solely intended to redirect signals onto high impedance input, such as COMP2 non-inverting input (highly resistive switch)

Bit 0 **COMP2EN**: Comparator 2 enable

This bit switches COMP2 ON/OFF.

0: Comparator 2 disabled

1: Comparator 2 enabled

17.5.3 COMP4 control and status register (COMP4_CSR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP4LOCK	COMP4OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP4_BLANKING[2:0]			COMP4HYST[1:0] ⁽¹⁾	
rw	r										rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP4POL	Res.	COMP4OUTSEL[3:0]				Res.	Res.	COMP4INPSEL ⁽¹⁾	COMP4INMSEL[2:0]			COMP4MODE[1:0] ⁽¹⁾		Res.	COMP4EN
rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		rw

1. Only in STM32F302xB/C.

Bit 31 **COMP4LOCK**: Comparator 4 lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

It allows to have COMP4_CSR register as read-only.

0: COMP4_CSR is read-write.

1: COMP4_CSR is read-only.

Bit 30 **COMP4OUT**: Comparator 4 output

This read-only bit is a copy of comparator 4 output state.

0: Output is low (non-inverting input below inverting input).

1: Output is high (non-inverting input above inverting input).

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

Bits 20:18 **COMP4_BLANKING**: Comparator 4 blanking source

These bits select which Timer output controls the comparator 4 output blanking.

000: No blanking

001: TIM3 OC4 selected as blanking source (STM32F302xB/C/D/E devices only)

010: Reserved

011: TIM15 OC1 selected as blanking source

Other configurations: reserved, must be kept at reset value

Bits 17:16 **COMP4HYST[1:0]**: Comparator 4 Hysteresis

On the STM32F302xBC, these bits control the hysteresis level.

00: No hysteresis

01: Low hysteresis

10: Medium hysteresis

11: High hysteresis

Please refer to the electrical characteristics for the hysteresis values.

On the STM32F302x6/8 and STM32F302xD/E, , these bits are reserved and must be kept at reset value.

Bit 15 **COMP4POL**: Comparator 4 output polarity

This bit is used to invert the comparator 4 output.

0: Output is not inverted

1: Output is inverted

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **COMP4OUTSEL[3:0]**: Comparator 4 output selection

These bits select which Timer input must be connected with the comparator4 output.

0000: No timer input selected
 0001: (BRK) Timer 1 break input
 0010: (BRK2) Timer 1 break input 2
 0011: Reserved
 0100: Reserved
 0101: Timer 1 break input 2
 0110: Timer 3 input capture 3
 0111: Reserved
 1000: Timer 15 input capture 2
 1001: Timer 4 input capture 2
 1010: Timer 15 OCREF_CLR input
 1011: Timer 3 OCrefclear input

Remaining combinations: reserved.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **COMP4INPSEL**: Comparator 4 non inverting input selection

0: PB0
 1: PE7

Note: On STM32F302x6/8 and STM32F302xD/E, this bit is reserved. COMP4_VINP is available on PB0 whatever value is written in bit 7.

Bits 6:4 **COMP4INMSEL[2:0]**: Comparator 4 inverting input selection

These bits allows to select the source connected to the inverting input of the comparator 4.

0000: 1/4 of Vrefint
 0001: 1/2 of Vrefint
 0010: 3/4 of Vrefint
 0011: Vrefint
 0100: PA4 or DAC1_CH1 output if enabled
 0101: Reserved
 0110: PE8
 0111: PB2

Remaining combinations: reserved.

Bits 3:2 **COMP4MODE[1:0]**: Comparator 1 mode (only in STM32F302xB/C devices)

These bits control the operating mode of the comparator 4 and allows to adjust the speed/consumption.

00: Ultra-low-power
 01: Low-power
 10: Medium speed
 11: High speed

Bit 1 Reserved, must be kept at reset value.

Bit 0 **COMP4EN**: Comparator 4 enable

This bit switches COMP4 ON/OFF.

0: Comparator 4 disabled
 1: Comparator 4 enabled

17.5.4 COMP6 control and status register (COMP6_CSR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP6LOCK	COMP6OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP6_BLANKING[2:0]			COMP6HYST[1:0] ⁽¹⁾	
rw	r										rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP6POL	Res.	COMP6OUTSEL[3:0]				Res.	Res.	COMP6INPSEL ⁽¹⁾	COMP6INMSEL[2:0]			COMP6MODE[1:0] ⁽¹⁾		Res.	COMP6EN
rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		rw

1. Only in STM32F302xB/C devices.

Bit 31 **COMP6LOCK**: Comparator 6 lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

It allows to have COMP6_CSR register as read-only.

0: COMP6_CSR is read-write.

1: COMP6_CSR is read-only.

Bit 30 **COMP6OUT**: Comparator 6 output

This read-only bit is a copy of comparator 6 output state.

0: Output is low (non-inverting input below inverting input).

1: Output is high (non-inverting input above inverting input).

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

Bits 20:18 **COMP6_BLANKING**: Comparator 6 blanking source

These bits select which Timer output controls the comparator 6 output blanking.

000: No blanking

001: Reserved

010: Reserved

011: TIM2 OC4 selected as blanking source

100: TIM15 OC2 selected as blanking source

Other configurations: reserved

The blanking signal is active high (masking comparator output signal). It is up to the user to program the comparator and blanking signal polarity correctly.

Bits 17:16 **COMP6HYST[1:0]**: Comparator 6 Hysteresis

On the STM32F302xBC, these bits control the hysteresis level.

00: No hysteresis

01: Low hysteresis

10: Medium hysteresis

11: High hysteresis

Please refer to the electrical characteristics for the hysteresis values.

On the STM32F302x6/8, these bits are reserved and must be kept at reset value.

Bit 15 **COMP6POL**: Comparator 6 output polarity

This bit is used to invert the comparator 6 output.

0: Output is not inverted

1: Output is inverted

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **COMP6OUTSEL[3:0]**: Comparator 6 output selection

These bits select which Timer input must be connected with the comparator 6 output.

- 0000: No timer input
- 0001: (BRK_ACTH) Timer 1 break input
- 0010: (BRK2) Timer 1 break input 2
- 0101: Timer 1 break input 2
- 0110: Timer 2 input capture 2
- 1000: Timer 2 OCREF_CLR input
- 1001: Timer 16 OCREF_CLR input
- 1010: Timer 16 input capture 1
- 1011: Timer 4 input capture 4

Remaining combinations: reserved.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **COMP6INPSEL**: Comparator 6 non inverting input selection

- 0: PD11
- 1: PB11

Note: On STM32F302x6/8 and STM32F302xD/E, this bit is reserved. COMP6_VINP is available on PB11 whatever value is written on bit 7.

Bits 6:4 **COMP6INMSEL[2:0]**: Comparator 6 inverting input selection

These bits allows to select the source connected to the inverting input of the comparator 6.

- 0000: 1/4 of Vrefint
- 0001: 1/2 of Vrefint
- 0010: 3/4 of Vrefint
- 0011: Vrefint
- 0100: PA4 or DAC1_CH1 output if enabled
- 0110: PD10
- 0111: PB15

Remaining combinations: reserved.

Bits 3:2 **COMP6MODE[1:0]**: Comparator 6 mode (only in STM32F302xB/C/D/E devices)

These bits control the operating mode of the comparator 6 and allows to adjust the speed/consumption.

- 00: Ultra-low-power
- 01: Low-power
- 10: Medium speed
- 11: High speed

Bit 1 Reserved, must be kept at reset value.

Bit 0 **COMP6EN**: Comparator 6 enable

This bit switches COMP6 ON/OFF.

- 0: Comparator 6 disabled
- 1: Comparator 6 enabled

17.5.5 COMP register map

The following table summarizes the comparator registers.

Table 104. COMP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	COMP1_CSR	COMP1LOCK	COMP1OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPx_BLANKING[2:0]	COMPx_BLANKING[2:0]	COMP1HYST[1:0]	COMP1POL	Res.	COMP1OUT SEL [3:0]	Res.	Res.	Res.	COMP1INSEL[2:0]	COMP1MODE[1:0]	COMP1_INP_DAC	COMP1EN								
	Reset value	0	0																							0	0	0	0	0	0	0	0
0x20	COMP2_CSR	COMP2LOCK	COMP2OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP2_BLANKING	COMP2POL	Res.	COMP2OUT SEL[3:0]	COMP2WINMODE	Res.	COMP2INSEL	COMP2INMSEL[2:0]	COMP2MODE[1:0]	COMP2_INP_DAC	COMP2EN										
	Reset value	0	0																					0	0	0	0	0	0	0	0	0	0
0x28	COMP4_CSR	COMP4LOCK	COMP4OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP4_BLANKING	COMP4POL	Res.	COMP4OUT SEL[3:0]	Res.	COMP4INSEL	COMP4INMSEL[2:0]	COMP4MODE[1:0]	Res.	COMP4EN											
	Reset value	0	0																				0	0	0	0	0	0	0	0	0	0	0
0x30	COMP6_CSR	COMP6LOCK	COMP6OUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP6_BLANKING	COMP6POL	Res.	COMP6OUT SEL[3:0]	Res.	COMP6INSEL	COMP6INMSEL[2:0]	COMP6MODE[1:0]	Res.	COMP6EN											
	Reset value	0	0																				0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

18 Operational amplifier (OPAMP)

18.1 OPAMP introduction

STM32F302xB/C/D/E devices embed 2 operational amplifiers OPAMP1, OPAMP2 and STM32F302x6/8 devices embed 1 operational amplifier OPAMP2. They can either be used as standalone amplifiers or as follower / programmable gain amplifiers.

The operational amplifier output is internally connected to an ADC channel for measurement purposes.

18.2 OPAMP main features

- Rail-to-rail input/output
- Low offset voltage
- Capability of being configured as a standalone operational amplifier or as a programmable gain amplifier (PGA)
- Access to all terminals
- Input multiplexer on inverting and non-inverting input
- Input multiplexer can be triggered by a timer and synchronized with a PWM signal.

18.3 OPAMP functional description

18.3.1 General description

On every OPAMP, there is one 4:1 multiplexer on the non-inverting input and one 2:1 multiplexer on the inverting input.

The inverting and non inverting inputs selection is made using the VM_SEL and VP_SEL bits respectively in the OPAMPx_CSR register.

The I/Os used as OPAMP input/outputs must be configured in analog mode in the GPIOs registers.

The connections with dedicated I/O are summarized in the table below and in [Figure 120](#) and [Figure 121](#).

Table 105. Connections with dedicated I/O

OPAMP1 inverting input ⁽¹⁾	OPAMP1 non inverting input ⁽¹⁾	OPAMP2 inverting input	OPAMP2 non inverting input
PA3 (VM1)	PA1 (VP0)	PA5 (VM1)	PA7 (VP0)
PC5 (VM0)	PA7 (VP1)	PC5 (VM0)	PB14 (VP1)
PA5 (VP3)	PA3 (VP2)	-	PB0 (VP2)
-	-	-	PD14 (VP3) ⁽¹⁾

1. Only in STM32F302xB/C/D/E devices.

18.3.2 Clock

The OPAMP clock provided by the clock controller is synchronized with the PCLK2 (APB2 clock). There is no clock enable control bit provided in the RCC controller. To use a clock source for the OPAMP, the SYSCFG clock enable control bit must be set in the RCC controller.

18.3.3 Operational amplifiers and comparators interconnections

Internal connections between the operational amplifiers and the comparators are useful in motor control applications. These connections are summarized in the following figures.

Figure 120. STM32F302xB/C/D/E comparator and operational amplifier connections

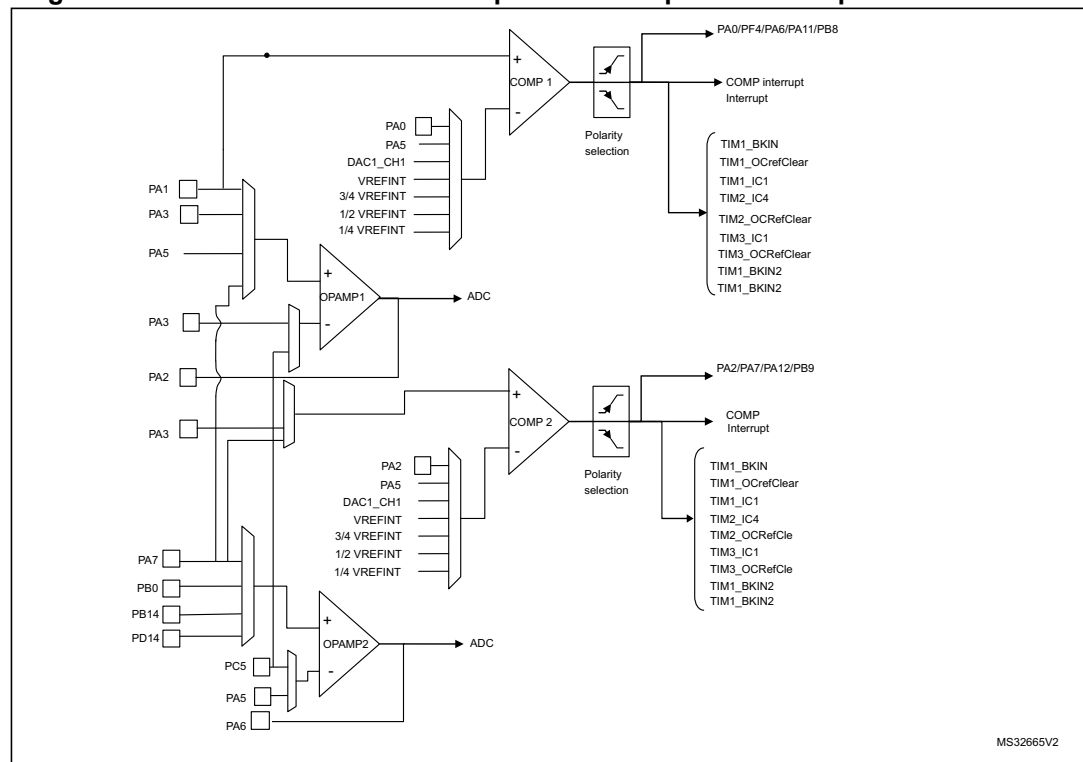
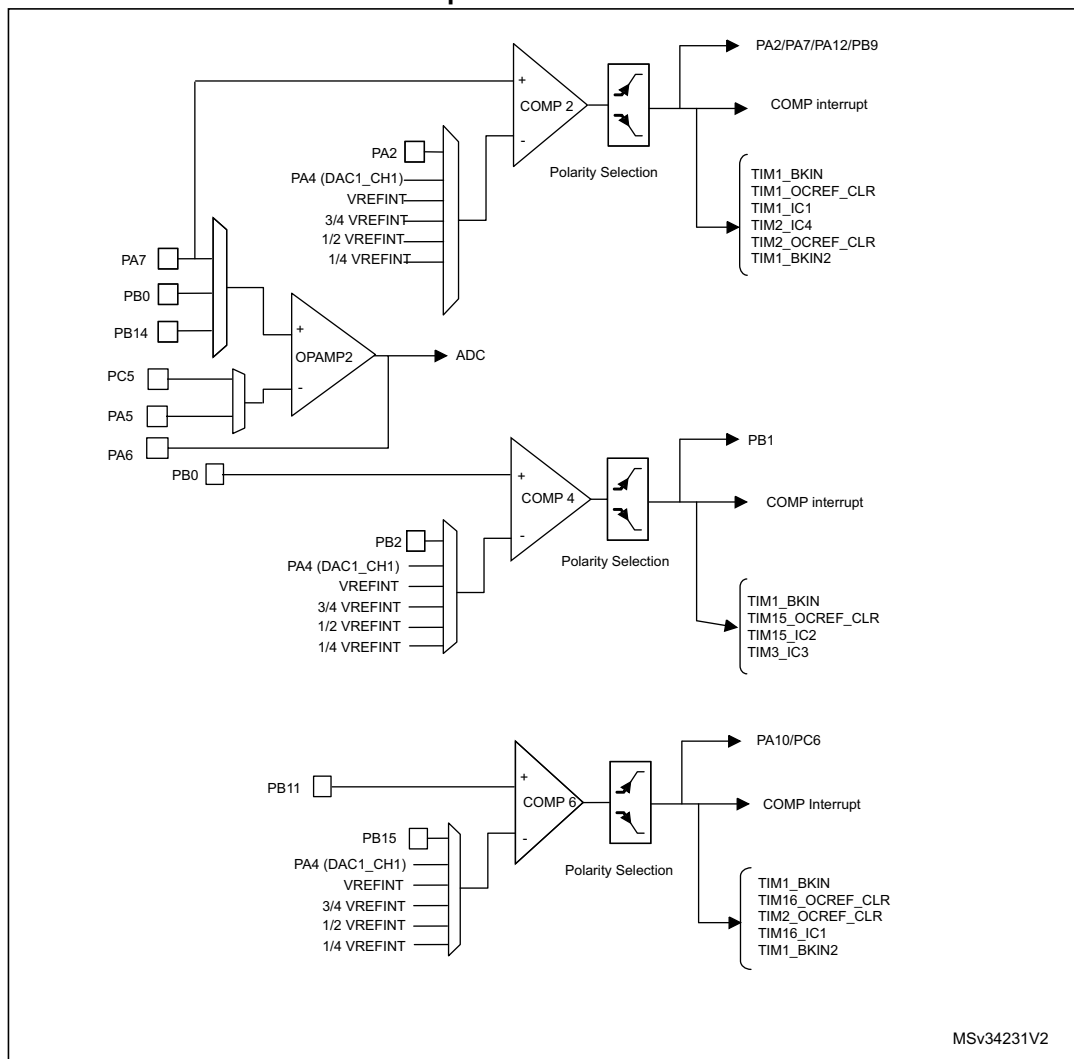


Figure 121. STM32F302x6/8 comparator and operational amplifier connections



18.3.4 Using the OPAMP outputs as ADC inputs

In order to use OPAMP outputs as ADC inputs, the operational amplifiers must be enabled and the ADC must use the OPAMP output channel number:

On the STM32F302xB/C:

- For OPAMP1, ADC1 channel 3 is used.
- For OPAMP2, ADC2 channel 3 is used.

On the STM32F302x6/8, only OPAMP2 is available and the ADC1 channel 10 is used on OPAMP2.

18.3.5 Calibration

The OPAMP interface continuously sends trimmed offset values to the 4 operational amplifiers. At startup, these values are initialized with the preset 'factory' trimming value.

Furthermore each operational amplifier offset can be trimmed by the user.

The user can switch from the 'factory' values to the 'user' trimmed values using the USER_TRIM bit in the OPAMP control register. This bit is reset at startup ('factory' values are sent to the operational amplifiers).

The rail-to-rail input stage of the OPAMP is composed of two differential pairs:

- One pair composed of NMOS transistors
- One pair composed of PMOS transistors.

As these two pairs are independent, the trimming procedure calibrates each one separately. The TRIMOFFSETN bits calibrate the NMOS differential pair offset and the TRIMOFFSETP bits calibrate the PMOS differential pair offset.

To calibrate the NMOS differential pair, the following conditions must be met: CALON=1 and CALSEL=11. In this case, an internal high voltage reference ($0.9 \times V_{DDA}$) is generated and applied on the inverting and non inverting OPAMP inputs connected together. The voltage applied to both inputs of the OPAMP can be measured (the OPAMP reference voltage can be output through the TSTREF bit and connected internally to an ADC channel; refer to [Section 15: Analog-to-digital converters \(ADC\) on page 286](#)). The software should increment the TRIMOFFSETN bits in the OPAMP control register from 0x00 to the first value that causes the OUTCAL bit to change from 1 to 0 in the OPAMP register. If the OUTCAL bit is reset, the offset is calibrated correctly and the corresponding trimming value must be stored.

The calibration of the PMOS differential pair is performed in the same way, with two differences: the TRIMOFFSETP bits-fields are used and the CALSEL bits must be programmed to '01' (an internal low voltage reference ($0.1 \times V_{DDA}$) is generated and applied on the inverting and non inverting OPAMP inputs connected together).

Note: *During calibration mode, to get the correct OUTCAL value, please make sure the OFFTRIMmax delay (specified in the datasheet electrical characteristics section) has elapsed between the write of a trimming value (TRIMOFFSETP or TRIMOFFSETN) and the read of the OUTCAL value,*

To calibrate the NMOS differential pair, use the following software procedure:

1. Enable OPAMP by setting the OPAMPxEN bit
2. Enable the user offset trimming by setting the USERTRIM bit
3. Connect VM and VP to the internal reference voltage by setting the CALON bit
4. Set CALSEL to 11 (OPAMP internal reference = $0.9 \times V_{DDA}$)
5. In a loop, increment the TRIMOFFSETN value. To exit from the loop, the OUTCAL bit must be reset. In this case, the TRIMOFFSETN value must be stored.

The same software procedure must be applied for PMOS differential pair calibration with CALSEL = 01 (OPAMP internal reference = $0.1 \times V_{DDA}$).

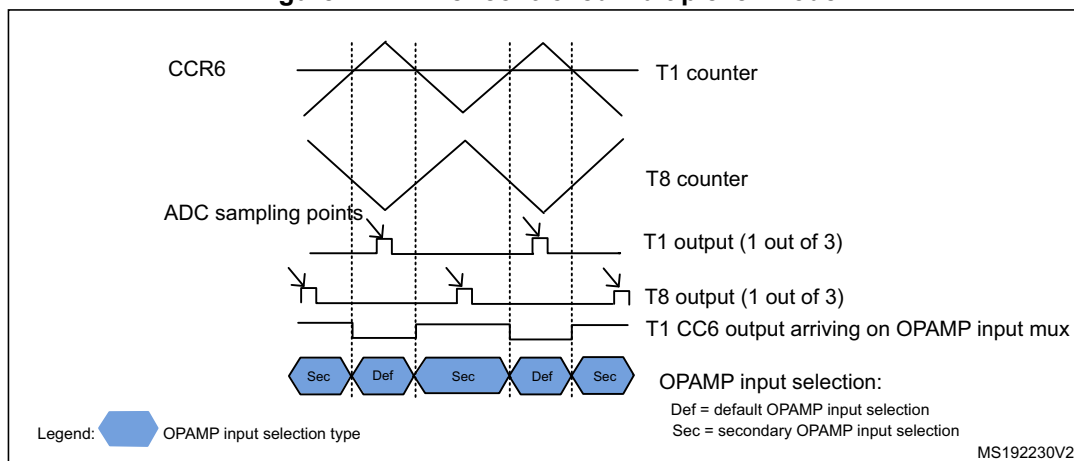
18.3.6 Timer controlled Multiplexer mode

The selection of the OPAMP inverting and non inverting inputs can be done automatically. In this case, the switch from one input to another is done automatically. This automatic switch is triggered by the TIM1 CC6 output arriving on the OPAMP input multiplexers.

This is useful for dual motor control with a need to measure the currents on the 3 phases instantaneously on a first motor and then on the second motor.

The automatic switch is enabled by setting the TCM_EN bit in the OPAMP control register. The inverting and non inverting inputs selection is performed using the VPS_SEL and VMS_SEL bit fields in the OPAMP control register. If the TCM_EN bit is cleared, the selection is done using the VP_SEL and VM_SEL bit fields in the OPAMP control register.

Figure 122. Timer controlled Multiplexer mode



18.3.7 OPAMP modes

The operational amplifier inputs and outputs are all accessible on terminals. The amplifiers can be used in multiple configuration environments:

- Standalone mode (external gain setting mode)
- Follower configuration mode
- PGA modes

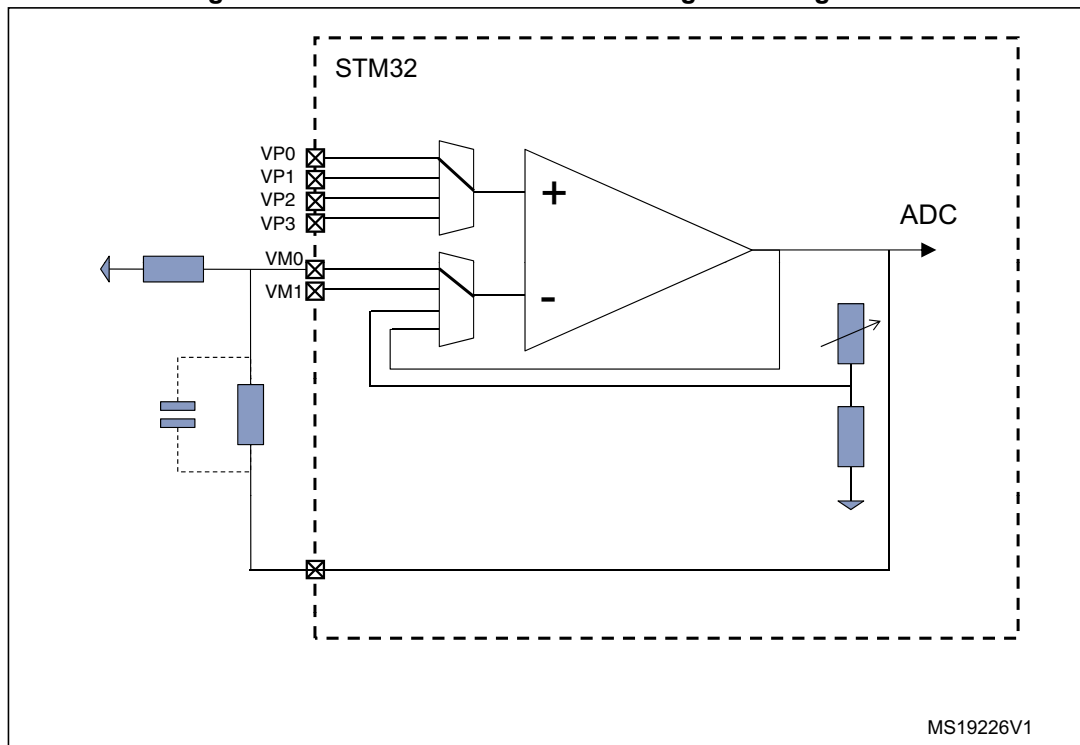
Important note: the amplifier output pin is directly connected to the output pad to minimize the output impedance. It cannot be used as a general purpose I/O, even if the amplifier is configured as a PGA and only connected to the ADC channel.

Note: *The impedance of the signal must be maintained below a level which avoids the input leakage to create significant artefacts (due to a resistive drop in the source). Please refer to the electrical characteristics section in the datasheet for further details.*

Standalone mode (external gain setting mode)

The external gain setting mode gives you full flexibility to choose the amplifier configuration and feedback networks. This mode is enabled by writing the VM_SEL bits in the OPAMPx_CR register to 00 or 01, to connect the inverting inputs to one of the two possible I/Os.

Figure 123. Standalone mode: external gain setting mode

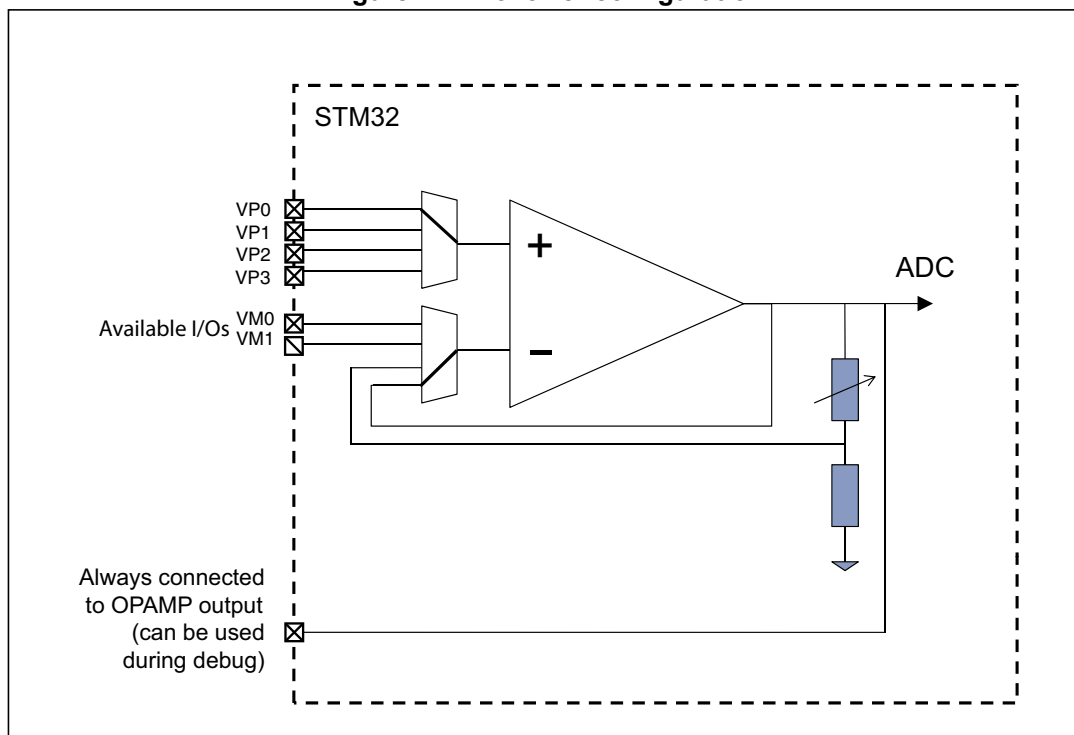


1. This figure gives an example in an inverting configuration. Any other option is possible, including comparator mode.

Follower configuration mode

The amplifier can be configured as a follower, by setting the VM_SEL bits to 11 in the OPAMPx_CR register. This allows you for instance to buffer signals with a relatively high impedance. In this case, the inverting inputs are free and the corresponding ports can be used as regular I/Os.

Figure 124. Follower configuration



1. This figure gives an example in an inverting configuration. Any other option is possible, including comparator mode.

Programmable Gain Amplifier mode

The Programmable Gain Amplifier (PGA) mode is enabled by writing the VM_SEL bits to 10 in the OPAMPx_CR register. The gain is set using the PGA_GAIN bits which must be set to 0x00..0x11 for gains ranging from 2 to 16.

In this case, the inverting inputs are internally connected to the central point of a built-in gain setting resistive network. [Figure 125: PGA mode, internal gain setting \(x2/x4/x8/x16\), inverting input not used](#) shows the internal connection in this mode.

An alternative option in PGA mode allows you to route the central point of the resistive network on one of the I/Os connected to the non-inverting input. This is enabled using the PGA_GAIN bits in OPAMPx_CR register:

- 10xx values are setting the gain and connect the central point to one of the two available inputs
- 11xx values are setting the gain and connect the central point to the second available input

This feature can be used for instance to add a low-pass filter to PGA, as shown in [Figure 126: PGA mode, internal gain setting \(x2/x4/x8/x16\), inverting input used for filtering](#). Please note that the cut-off frequency is changed if the gain is modified (refer to the electrical characteristics section of the datasheet for details on resistive network elements).

Figure 125. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used

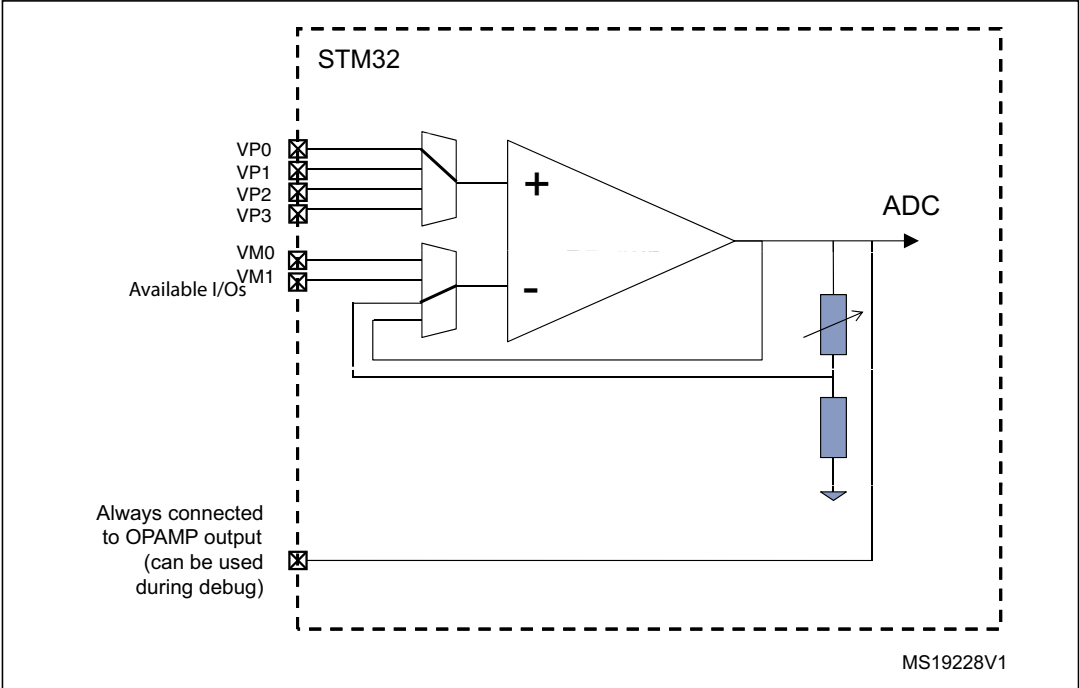
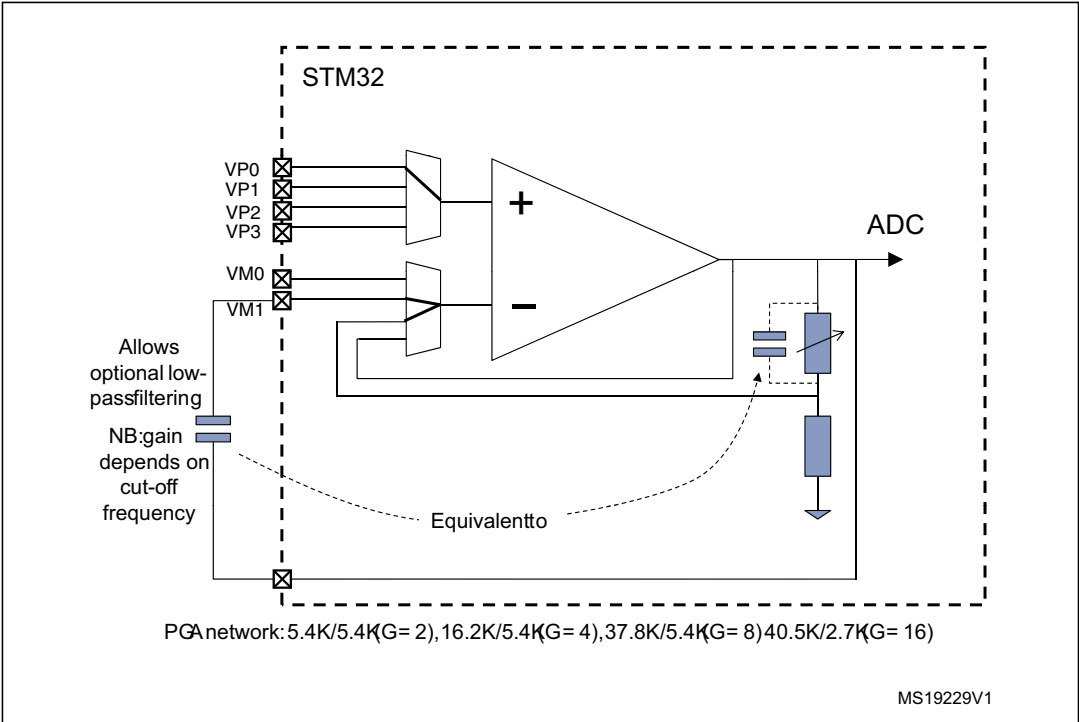


Figure 126. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering



18.4 OPAMP registers

18.4.1 OPAMP1 control register (OPAMP1_CSR)

Note: This register is only available in STM32F302xB/C/D/E devices.

Address offset : 0x38

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OUTCAL	TSTREF	TRIMOFFSETN					TRIMOFFSETP					USER_TRIM	PGA_GAIN	
rw	r	rw	rw					rw					rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN		CALSEL		CALON	VPS_SEL		VMS_SEL	TCM_EN	VM_SEL		Res.	VP_SEL		FORCE_VP	OPAMP1EN
rw		rw		rw	rw		rw	rw	rw			rw		rw	rw

Bit 31 **LOCK:** OPAMP 1 lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP1_CSR register as read-only.

0: OPAMP1_CSR is read-write.

1: OPAMP1_CSR is read-only.

Bit 30 **OUTCAL:**

OPAMP output status flag, when the OPAMP is used as comparator during calibration.

0: Non-inverting < inverting

1: Non-inverting > inverting.

Bit 29 **TSTREF:**

This bit is set and cleared by software. It is used to output the internal reference voltage ($V_{REFOPAMP1}$).

0: $V_{REFOPAMP1}$ is output.

1: $V_{REFOPAMP1}$ is not output.

Bits 28:24 **TRIMOFFSETN:** Offset trimming value (NMOS)

Bits 23:19 **TRIMOFFSETP:** Offset trimming value (PMOS)

Bit 18 **USER_TRIM:** User trimming enable.

This bit is used to configure the OPAMP offset.

0: User trimming disabled.

1: User trimming enabled.

Bits 17:14 **PGA_GAIN**: Gain in PGA mode

0X00 = Non-inverting gain = 2
 0X01 = Non-inverting gain = 4
 0X10 = Non-inverting gain = 8
 0X11 = Non-inverting gain = 16
 1000 = Non-inverting gain = 2 - Internal feedback connected to VM0
 1001 = Non-inverting gain = 4 - Internal feedback connected to VM0
 1010 = Non-inverting gain = 8 - Internal feedback connected to VM0
 1011 = Non-inverting gain = 16 - Internal feedback connected to VM0
 1100 = Non-inverting gain = 2 - Internal feedback connected to VM1
 1101 = Non-inverting gain = 4 - Internal feedback connected to VM1
 1110 = Non-inverting gain = 8 - Internal feedback connected to VM1
 1111 = Non-inverting gain = 16 - Internal feedback connected to VM1

Bits 13:12 **CALSEL**: Calibration selection

This bit is set and cleared by software. It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP = 1.

00 = $V_{REFOPAMP} = 3.3\% V_{DDA}$
 01 = $V_{REFOPAMP} = 10\% V_{DDA}$
 10 = $V_{REFOPAMP} = 50\% V_{DDA}$
 11 = $V_{REFOPAMP} = 90\% V_{DDA}$

Bit 11 **CALON**: Calibration mode enable

This bit is set and cleared by software. It is used to enable the calibration mode connecting VM and VP to the OPAMP internal reference voltage.

0: Calibration mode disabled.
 1: Calibration mode enabled.

Bits 10:9 **VPS_SEL**: OPAMP1 Non inverting input secondary selection.

These bits are set and cleared by software. They are used to select the OPAMP1 non inverting input when TCM_EN = 1.

00: PA7 used as OPAMP1 non inverting input
 01: PA5 used as OPAMP1 non inverting input
 10: PA3 used as OPAMP1 non inverting input
 11: PA1 used as OPAMP1 non inverting input

Bit 8 **VMS_SEL**: OPAMP1 inverting input secondary selection

This bit is set and cleared by software. It is used to select the OPAMP1 inverting input when TCM_EN = 1.

0: PC5 (VM0) used as OPAMP1 inverting input
 1: PA3 (VM1) used as OPAMP1 inverting input

Bit 7 **TCM_EN**: Timer controlled Mux mode enable

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs.

Bits 6:5 **VM_SEL**: OPAMP1 inverting input selection.

These bits are set and cleared by software. They are used to select the OPAMP1 inverting input.

00: PC5 (VM0) used as OPAMP1 inverting input
 01: PA3 (VM1) used as OPAMP1 inverting input
 10: Resistor feedback output (PGA mode)
 11: follower mode

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **VP_SEL**: OPAMP1 Non inverting input selection.

These bits are set and cleared by software. They are used to select the OPAMP1 non inverting input.

00: PA7 used as OPAMP1 non inverting input

01: PA5 used as OPAMP1 non inverting input

10: PA3 used as OPAMP1 non inverting input

11: PA1 used as OPAMP1 non inverting input

Bit 1 **FORCE_VP**:

This bit forces a calibration reference voltage on non-inverting input and disables external connections.

0: Normal operating mode. Non-inverting input connected to inputs.

1: Calibration mode. Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAMP1EN**: OPAMP1 enable.

This bit is set and cleared by software. It is used to enable the OPAMP1.

0: OPAMP1 is disabled.

1: OPAMP1 is enabled.

18.4.2 OPAMP2 control register (OPAMP2_CSR)

Address offset: 0x3C

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OUTCAL	TSTREF	TRIMOFFSETN					TRIMOFFSETP					USER_TRIM	PGA_GAIN	
rw	r	rw	rw					rw					rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN		CALSEL		CALON	VPS_SEL		VMS_SEL	TCM_EN	VM_SEL		Res.	VP_SEL		FORCE_VP	OPAMP2EN
rw		rw		rw	rw		rw	rw	rw			rw		rw	rw

Bit 31 **LOCK**: OPAMP 2 lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

This bit is used to configure the OPAMP2_CSR register as read-only.

0: OPAMP2_CSR is read-write.

1: OPAMP2_CSR is read-only.

Bit 30 **OUTCAL**:

OPAMP output status flag, when the OPAMP is used as comparator during calibration.

0: Non-inverting < inverting

1: Non-inverting > inverting.

Bit 29 **TSTREF**:

This bit is set and cleared by software. It is used to output the internal reference voltage ($V_{REFOPAMP2}$).

0: $V_{REFOPAMP2}$ is output.

1: $V_{REFOPAMP2}$ is not output.

Bits 28:24 **TRIMOFFSETN**: Offset trimming value (NMOS)

Bits 23:19 **TRIMOFFSETP**: Offset trimming value (PMOS)

Bit 18 **USER_TRIM**: User trimming enable.

This bit is used to configure the OPAMP offset.

0: User trimming disabled.

1: User trimming enabled.

Bits 17:14 **PGA_GAIN**: gain in PGA mode

0X00 = Non-inverting gain = 2

0X01 = Non-inverting gain = 4

0X10 = Non-inverting gain = 8

0X11 = Non-inverting gain = 16

1000 = Non-inverting gain = 2 - Internal feedback connected to VM0

1001 = Non-inverting gain = 4 - Internal feedback connected to VM0

1010 = Non-inverting gain = 8 - Internal feedback connected to VM0

1011 = Non-inverting gain = 16 - Internal feedback connected to VM0

1100 = Non-inverting gain = 2 - Internal feedback connected to VM1

1101 = Non-inverting gain = 4 - Internal feedback connected to VM1

1110 = Non-inverting gain = 8 - Internal feedback connected to VM1

1111 = Non-inverting gain = 16 - Internal feedback connected to VM1

Bits 13:12 **CALSEL**: Calibration selection

This bit is set and cleared by software. It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP = 1.

00 = $V_{REFOPAMP} = 3.3\% V_{DDA}$

01 = $V_{REFOPAMP} = 10\% V_{DDA}$

10 = $V_{REFOPAMP} = 50\% V_{DDA}$

11 = $V_{REFOPAMP} = 90\% V_{DDA}$

Bit 11 **CALON**: Calibration mode enable

This bit is set and cleared by software. It is used to enable the calibration mode connecting VM and VP to the OPAMP internal reference voltage.

0: calibration mode disabled.

1: calibration mode enabled.

Bits 10:9 **VPS_SEL**: OPAMP2 Non inverting input secondary selection.

These bits are set and cleared by software. They are used to select the OPAMP2 non inverting input when TCM_EN = 1.

00: PD14 used as OPAMP2 non inverting input (STM32F302xB/C/D/E devices only)

01: PB14 used as OPAMP2 non inverting input

10: PB0 used as OPAMP2 non inverting input

11: PA7 used as OPAMP2 non inverting input

Bit 8 **VMS_SEL**: OPAMP2 inverting input secondary selection

This bit is set and cleared by software. It is used to select the OPAMP2 inverting input when TCM_EN = 1.

0: PC5 (VM0) used as OPAMP2 inverting input

1: PA5 (VM1) used as OPAMP2 inverting input

Bit 7 **TCM_EN**: Timer controlled Mux mode enable.

This bit is set and cleared by software. It is used to control automatically the switch between the default selection (VP_SEL and VM_SEL) and the secondary selection (VPS_SEL and VMS_SEL) of the inverting and non inverting inputs.

Bit 6:5 **VM_SEL**: OPAMP2 inverting input selection.

These bits are set and cleared by software. They are used to select the OPAMP2 inverting input.

00: PC5 (VM0) used as OPAMP2 inverting input

01: PA5 (VM1) used as OPAMP2 inverting input

10: Resistor feedback output (PGA mode)

11: follower mode

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **VP_SEL**: OPAMP2 non inverting input selection.

These bits are set/reset by software. They are used to select the OPAMP2 non inverting input.

00: PD14 used as OPAMP2 non inverting input (STM32F302xB/C/D/E devices only)

01: PB14 used as OPAMP2 non inverting input

10: PB0 used as OPAMP2 non inverting input

11: PA7 used as OPAMP2 non inverting input

Bit 1 **FORCE_VP**:

This bit forces a calibration reference voltage on non-inverting input and disables external connections.

0: Normal operating mode. Non-inverting input connected to inputs.

1: Calibration mode. Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAMP2EN**: OPAMP2 enable.

This bit is set and cleared by software. It is used to select the OPAMP2.

0: OPAMP2 is disabled.

1: OPAMP2 is enabled.

18.4.3 OPAMP register map

The following table summarizes the OPAMP registers.

Table 106. OPAMP 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
0x38	OPAMP1_CSR	LOCK	OUTCAL	TSTREF			TRIMOFFSETN					TRIMOFFSETP			USER_TRIM		PGA_GAIN			CALSEL		CALON	VPS_SEL	VMS_SEL	TCM_EN	VM_SEL		Res	VP_SEL	FORCE_VP	OPAMP1EN		
	Reset value	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	0	0		0	0	0	0
0x3C	OPAMP2_CSR	LOCK	OUTCAL	TSTREF			TRIMOFFSETN					TRIMOFFSETP			USER_TRIM		PGA_GAIN			CALSEL		CALON	VPS_SEL	VMS_SEL	TCM_EN	VM_SEL		Res	VP_SEL	FORCE_VP	OPAMP2EN		
	Reset value	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	0	0		0	0	0	0

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

19 Touch sensing controller (TSC)

19.1 Introduction

The touch sensing controller provides a simple solution for adding capacitive sensing functionality to any application. Capacitive sensing technology is able to detect finger presence near an electrode which is protected from direct touch by a dielectric (for example glass, plastic). The capacitive variation introduced by the finger (or any conductive object) is measured using a proven implementation based on a surface charge transfer acquisition principle.

The touch sensing controller is fully supported by the STMTouch touch sensing firmware library which is free to use and allows touch sensing functionality to be implemented reliably in the end application.

19.2 TSC main features

The touch sensing controller has the following main features:

- Proven and robust surface charge transfer acquisition principle
- Supports up to 24 capacitive sensing channels on STM32F302xBxC devices and 18 on STM32F302x6/x8 devices
- Up to 8 capacitive sensing channels on STM32F302xBxC devices and 6 on STM32F302x6/x8 devices can be acquired in parallel offering a very good response time
- Spread spectrum feature to improve system robustness in noisy environments
- full hardware management of the charge transfer acquisition sequence
- Programmable charge transfer frequency
- Programmable sampling capacitor I/O pin
- Programmable channel I/O pin
- Programmable max count value to avoid long acquisition when a channel is faulty
- Dedicated end of acquisition and max count error flags with interrupt capability
- One sampling capacitor for up to 3 capacitive sensing channels to reduce the system components
- Compatible with proximity, touchkey, linear and rotary touch sensor implementation
- Designed to operate with STMTouch touch sensing firmware library

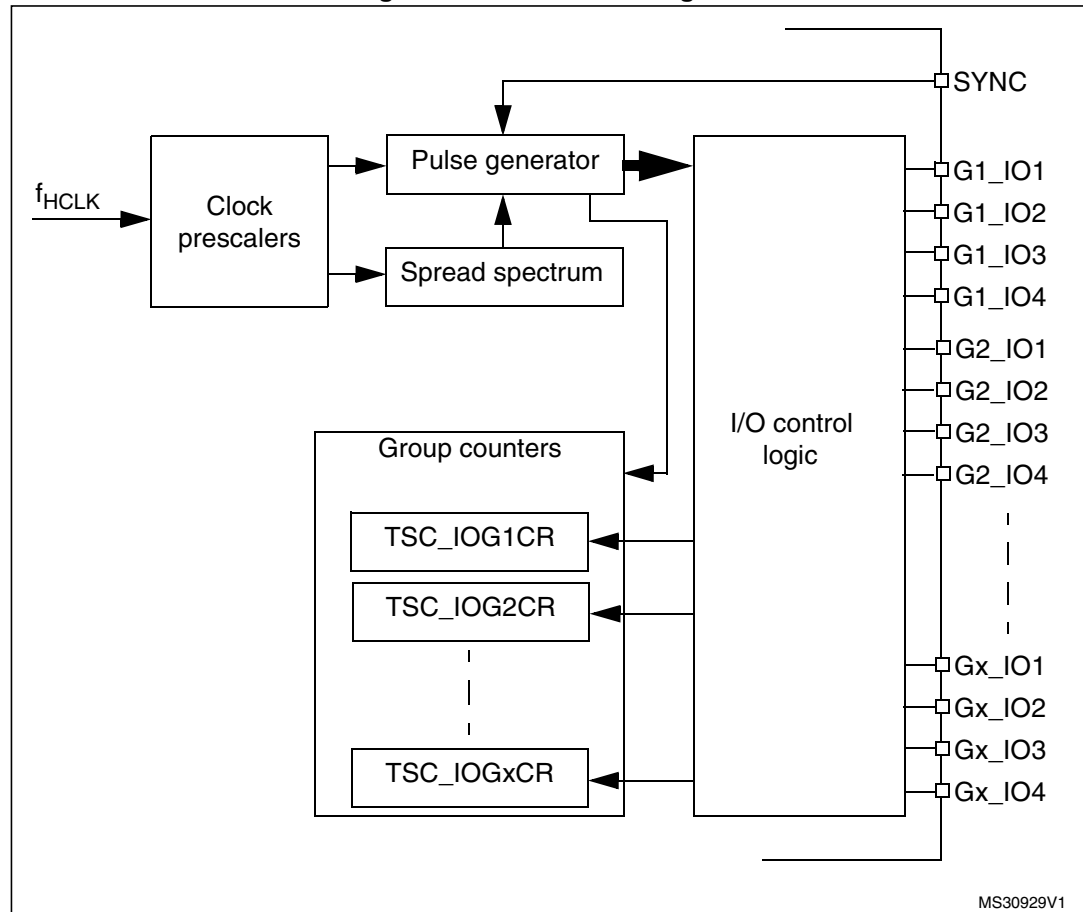
Note: The number of capacitive sensing channels is dependent on the size of the packages and subject to IO availability.

19.3 TSC functional description

19.3.1 TSC block diagram

The block diagram of the touch sensing controller is shown in [Figure 127: TSC block diagram](#).

Figure 127. TSC block diagram



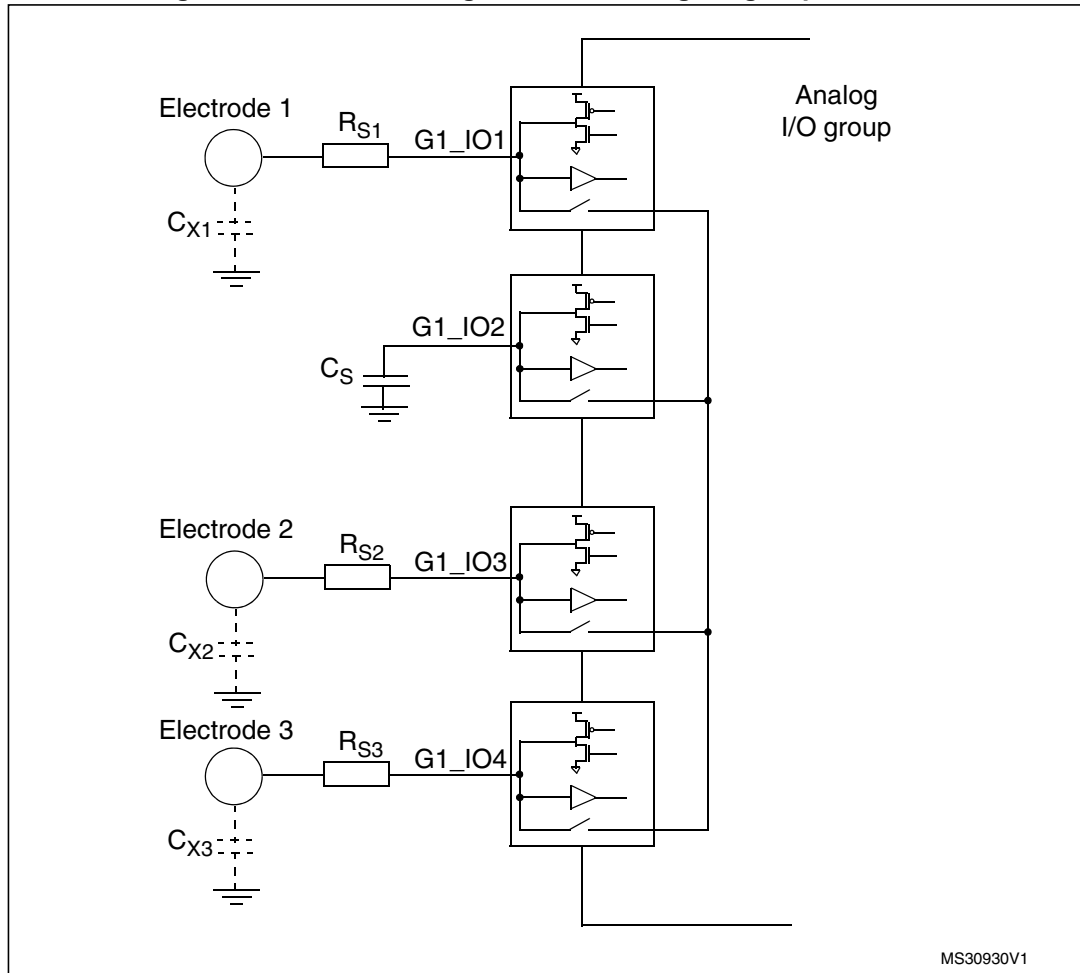
19.3.2 Surface charge transfer acquisition overview

The surface charge transfer acquisition is a proven, robust and efficient way to measure a capacitance. It uses a minimum number of external components to operate with a single ended electrode type. This acquisition is designed around an analog I/O group which is composed of four GPIOs (see [Figure 128](#)). Several analog I/O groups are available to allow the acquisition of several capacitive sensing channels simultaneously and to support a larger number of capacitive sensing channels. Within a same analog I/O group, the acquisition of the capacitive sensing channels is sequential.

One of the GPIOs is dedicated to the sampling capacitor C_S . Only one sampling capacitor I/O per analog I/O group must be enabled at a time.

The remaining GPIOs are dedicated to the electrodes and are commonly called channels. For some specific needs (such as proximity detection), it is possible to simultaneously enable more than one channel per analog I/O group.

Figure 128. Surface charge transfer analog I/O group structure



Note: Gx_IOy where x is the analog I/O group number and y the GPIO number within the selected group.

The surface charge transfer acquisition principle consists of charging an electrode capacitance (C_X) and transferring a part of the accumulated charge into a sampling capacitor (C_S). This sequence is repeated until the voltage across C_S reaches a given threshold (V_{IH} in our case). The number of charge transfers required to reach the threshold is a direct representation of the size of the electrode capacitance.

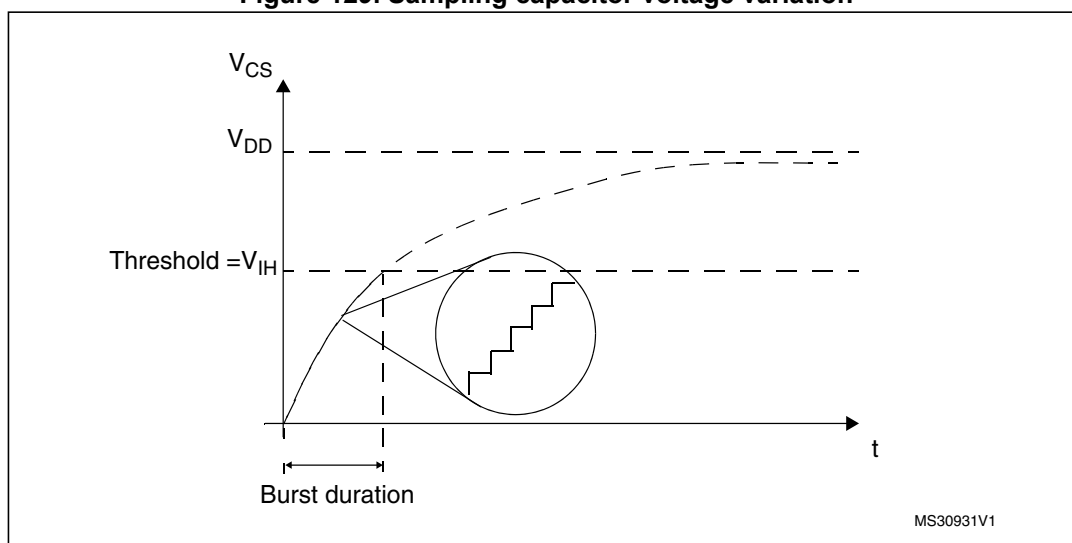
The [Table 107](#) details the charge transfer acquisition sequence of the capacitive sensing channel 1. States 3 to 7 are repeated until the voltage across C_S reaches the given threshold. The same sequence applies to the acquisition of the other channels. The electrode serial resistor R_S improves the ESD immunity of the solution.

Table 107. Acquisition sequence summary

State	G1_IO1 (electrode)	G1_IO2 (sampling)	G1_IO3 (electrode)	G1_IO4 (electrode)	State description
#1	Input floating with analog switch closed	Output open-drain low with analog switch closed	Input floating with analog switch closed		Discharge all C_X and C_S
#2	Input floating				Dead time
#3	Output push-pull high	Input floating			Charge C_{X1}
#4	Input floating				Dead time
#5	Input floating with analog switch closed		Input floating		Charge transfer from C_{X1} to C_S
#6	Input floating				Dead time
#7	Input floating				Measure C_S voltage

The voltage variation over the time on the sampling capacitor C_S is detailed below:

Figure 129. Sampling capacitor voltage variation



19.3.3 Reset and clocks

The TSC clock source is the AHB clock (HCLK). Two programmable prescalers are used to generate the pulse generator and the spread spectrum internal clocks:

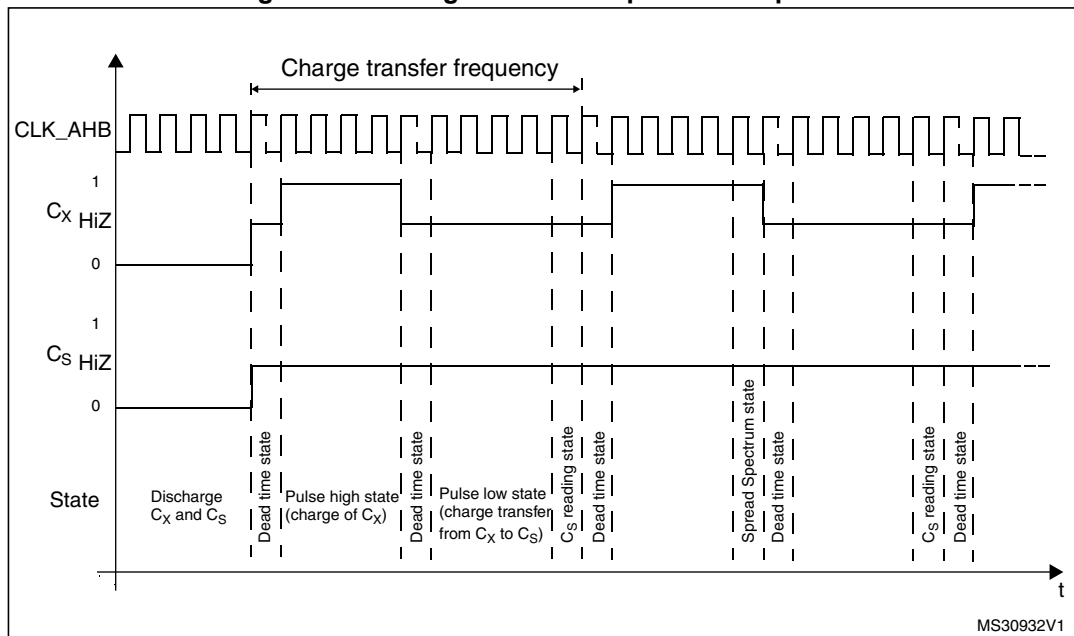
- The pulse generator clock (PGCLK) is defined using the PGPSC[2:0] bits of the TSC_CR register
- The spread spectrum clock (SSCLK) is defined using the SSPSC bit of the TSC_CR register

The Reset and Clock Controller (RCC) provides dedicated bits to enable the touch sensing controller clock and to reset this peripheral. For more information, please refer to [Section 10: Reset and clock control \(RCC\)](#).

19.3.4 Charge transfer acquisition sequence

An example of a charge transfer acquisition sequence is detailed in [Figure 130](#).

Figure 130. Charge transfer acquisition sequence



For higher flexibility, the charge transfer frequency is fully configurable. Both the pulse high state (charge of C_x) and the pulse low state (transfer of charge from C_x to C_s) duration can be defined using the CTPH[3:0] and CTPL[3:0] bits in the TSC_CR register. The standard range for the pulse high and low states duration is 500 ns to 2 μ s. To ensure a correct measurement of the electrode capacitance, the pulse high state duration must be set to ensure that C_x is always fully charged.

A dead time where both the sampling capacitor I/O and the channel I/O are in input floating state is inserted between the pulse high and low states to ensure an optimum charge transfer acquisition sequence. This state duration is 2 periods of HCLK.

At the end of the pulse high state and if the spread spectrum feature is enabled, a variable number of periods of the SSCLK clock are added.

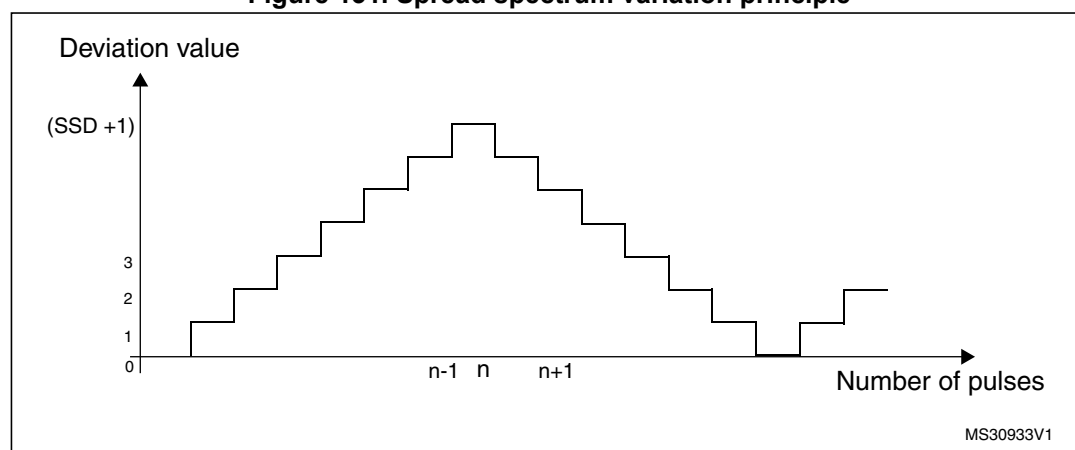
The reading of the sampling capacitor I/O, to determine if the voltage across C_s has reached the given threshold, is performed at the end of the pulse low state and its duration is one period of HCLK.

19.3.5 Spread spectrum feature

The spread spectrum feature allows to generate a variation of the charge transfer frequency. This is done to improve the robustness of the charge transfer acquisition in noisy environments and also to reduce the induced emission. The maximum frequency variation is in the range of 10% to 50% of the nominal charge transfer period. For instance, for a nominal charge transfer frequency of 250 kHz (4 μ s), the typical spread spectrum deviation is 10% (400 ns) which leads to a minimum charge transfer frequency of ~227 kHz.

In practice, the spread spectrum consists of adding a variable number of SSCLK periods to the pulse high state using the principle shown below:

Figure 131. Spread spectrum variation principle



The table below details the maximum frequency deviation with different HCLK settings:

Table 108. Spread spectrum deviation versus AHB clock frequency

f_{HCLK}	Spread spectrum step	Maximum spread spectrum deviation
24 MHz	41.6 ns	10666.6 ns
48 MHz	20.8 ns	5333.3 ns

The spread spectrum feature can be disabled/enabled using the SSE bit in the TSC_CR register. The frequency deviation is also configurable to accommodate the device HCLK clock frequency and the selected charge transfer frequency through the SSPSC and SSD[6:0] bits in the TSC_CR register.

19.3.6 Max count error

The max count error prevents long acquisition times resulting from a faulty capacitive sensing channel. It consists of specifying a maximum count value for the analog I/O group counters. This maximum count value is specified using the MCV[2:0] bits in the TSC_CR register. As soon as an acquisition group counter reaches this maximum value, the ongoing acquisition is stopped and the end of acquisition (EOAF bit) and max count error (MCEF bit) flags are both set. An interrupt can also be generated if the corresponding end of acquisition (EOAIE bit) or/and max count error (MCEIE bit) interrupt enable bits are set.

19.3.7 Sampling capacitor I/O and channel I/O mode selection

To allow the GPIOs to be controlled by the touch sensing controller, the corresponding alternate function must be enabled through the standard GPIO registers and the GPIOxAFR registers.

The GPIOs modes controlled by the TSC are defined using the TSC_IOSCR and TSC_IOCCR register.

When there is no ongoing acquisition, all the I/Os controlled by the touch sensing controller are in default state. While an acquisition is ongoing, only unused I/Os (neither defined as sampling capacitor I/O nor as channel I/O) are in default state. The IODEF bit in the TSC_CR register defines the configuration of the I/Os which are in default state. The table below summarizes the configuration of the I/O depending on its mode.

Table 109. I/O state depending on its mode and IODEF bit value

IODEF bit	Acquisition status	Unused I/O mode	Electrode I/O mode	Sampling capacitor I/O mode
0 (output push-pull low)	No	Output push-pull low	Output push-pull low	Output push-pull low
0 (output push-pull low)	ongoing	Output push-pull low	-	-
1 (input floating)	No	Input floating	Input floating	Input floating
1 (input floating)	ongoing	Input floating	-	-

Unused I/O mode

An unused I/O corresponds to a GPIO controlled by the TSC peripheral but not defined as an electrode I/O nor as a sampling capacitor I/O.

Sampling capacitor I/O mode

To allow the control of the sampling capacitor I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output open drain mode and then the corresponding Gx_IOy bit in the TSC_IOSCR register must be set.

Only one sampling capacitor per analog I/O group must be enabled at a time.

Channel I/O mode

To allow the control of the channel I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output push-pull mode and the corresponding Gx_IOy bit in the TSC_IOCCR register must be set.

For proximity detection where a higher equivalent electrode surface is required or to speed-up the acquisition process, it is possible to enable and simultaneously acquire several channels belonging to the same analog I/O group.

Note: During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR or TSC_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

19.3.8 Acquisition mode

The touch sensing controller offers two acquisition modes:

- Normal acquisition mode: the acquisition starts as soon as the START bit in the TSC_CR register is set.
- Synchronized acquisition mode: the acquisition is enabled by setting the START bit in the TSC_CR register but only starts upon the detection of a falling edge or a rising edge and high level on the SYNC input pin. This mode is useful for synchronizing the capacitive sensing channels acquisition with an external signal without additional CPU load.

The GxE bits in the TSC_I OGCSR registers specify which analog I/O groups are enabled (corresponding counter is counting). The C_S voltage of a disabled analog I/O group is not monitored and this group does not participate in the triggering of the end of acquisition flag. However, if the disabled analog I/O group contains some channels, they will be pulsed.

When the C_S voltage of an enabled analog I/O group reaches the given threshold, the corresponding GxS bit of the TSC_I OGCSR register is set. When the acquisition of all enabled analog I/O groups is complete (all GxS bits of all enabled analog I/O groups are set), the EOAF flag in the TSC_ISR register is set. An interrupt request is generated if the EOAI E bit in the TSC_I ER register is set.

In the case that a max count error is detected, the ongoing acquisition is stopped and both the EOAF and MCEF flags in the TSC_ISR register are set. Interrupt requests can be generated for both events if the corresponding bits (EOAI E and MCEI E bits of the TSCIER register) are set. Note that when the max count error is detected the remaining GxS bits in the enabled analog I/O groups are not set.

To clear the interrupt flags, the corresponding EOAI C and MCEI C bits in the TSC_I CR register must be set.

The analog I/O group counters are cleared when a new acquisition is started. They are updated with the number of charge transfer cycles generated on the corresponding channel(s) upon the completion of the acquisition.

19.3.9 I/O hysteresis and analog switch control

In order to offer a higher flexibility, the touch sensing controller also allows to take the control of the Schmitt trigger hysteresis and analog switch of each Gx_I Oy. This control is available whatever the I/O control mode is (controlled by standard GPIO registers or other peripherals) assuming that the touch sensing controller is enabled. This may be useful to perform a different acquisition sequence or for other purposes.

In order to improve the system immunity, the Schmitt trigger hysteresis of the GPIOs controlled by the TSC must be disabled by resetting the corresponding Gx_I Oy bit in the TSC_I OHCR register.

19.3.10 Capacitive sensing GPIOs

The table below provides an overview of the capacitive sensing GPIOs.

19.4 TSC low-power modes

Table 110. Effect of low-power modes on TSC

Mode	Description
Sleep	No effect TSC interrupts cause the device to exit Sleep mode.
Stop	TSC registers are frozen
Standby	The TSC stops its operation until the Stop or Standby mode is exited.

19.5 TSC interrupts

Table 111. Interrupt control bits

Interrupt event	Enable control bit	Event flag	Clear flag bit	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
End of acquisition	EOAIE	EOAIF	EOAIC	yes	no	no
Max count error	MCEIE	MCEIF	MCEIC	yes	no	no

19.6 TSC registers

Refer to [Section 2.1 on page 42](#) of the reference manual for a list of abbreviations used in register descriptions.

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

19.6.1 TSC control register (TSC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTPH[3:0]				CTPL[3:0]				SSD[6:0]						SSE	
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
SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV[2:0]			IODEF	SYNC POL	AM	START	TSCE
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **CTPH[3:0]**: Charge transfer pulse high

These bits are set and cleared by software. They define the duration of the high state of the charge transfer pulse (charge of C_X).

0000: $1 \times t_{PGCLK}$

0001: $2 \times t_{PGCLK}$

...

1111: $16 \times t_{PGCLK}$

Note: These bits must not be modified when an acquisition is ongoing.

Bits 27:24 **CTPL[3:0]**: Charge transfer pulse low

These bits are set and cleared by software. They define the duration of the low state of the charge transfer pulse (transfer of charge from C_X to C_S).

0000: $1 \times t_{PGCLK}$

0001: $2 \times t_{PGCLK}$

...

1111: $16 \times t_{PGCLK}$

Note: These bits must not be modified when an acquisition is ongoing.

Bits 23:17 **SSD[6:0]**: Spread spectrum deviation

These bits are set and cleared by software. They define the spread spectrum deviation which consists in adding a variable number of periods of the SSCLK clock to the charge transfer pulse high state.

0000000: $1 \times t_{SSCLK}$

0000001: $2 \times t_{SSCLK}$

...

1111111: $128 \times t_{SSCLK}$

Note: These bits must not be modified when an acquisition is ongoing.

Bit 16 **SSE**: Spread spectrum enable

This bit is set and cleared by software to enable/disable the spread spectrum feature.

0: Spread spectrum disabled

1: Spread spectrum enabled

Note: This bit must not be modified when an acquisition is ongoing.

Bit 15 **SSPSC**: Spread spectrum prescaler

This bit is set and cleared by software. It selects the AHB clock divider used to generate the spread spectrum clock (SSCLK).

0: f_{HCLK}

1: $f_{HCLK} / 2$

Note: This bit must not be modified when an acquisition is ongoing.

Bits 14:12 **PGPSC[2:0]**: pulse generator prescaler

These bits are set and cleared by software. They select the AHB clock divider used to generate the pulse generator clock (PGCLK).

000: f_{HCLK}

001: $f_{HCLK} / 2$

010: $f_{HCLK} / 4$

011: $f_{HCLK} / 8$

100: $f_{HCLK} / 16$

101: $f_{HCLK} / 32$

110: $f_{HCLK} / 64$

111: $f_{HCLK} / 128$

Note: These bits must not be modified when an acquisition is ongoing.

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:5 **MCV[2:0]**: Max count value

These bits are set and cleared by software. They define the maximum number of charge transfer pulses that can be generated before a max count error is generated.

000: 255
 001: 511
 010: 1023
 011: 2047
 100: 4095
 101: 8191
 110: 16383
 111: reserved

Note: These bits must not be modified when an acquisition is ongoing.

Bit 4 **IODEF**: I/O Default mode

This bit is set and cleared by software. It defines the configuration of all the TSC I/Os when there is no ongoing acquisition. When there is an ongoing acquisition, it defines the configuration of all unused I/Os (not defined as sampling capacitor I/O or as channel I/O).

0: I/Os are forced to output push-pull low
 1: I/Os are in input floating

Note: This bit must not be modified when an acquisition is ongoing.

Bit 3 **SYNCPOL**: Synchronization pin polarity

This bit is set and cleared by software to select the polarity of the synchronization input pin.

0: Falling edge only
 1: Rising edge and high level

Bit 2 **AM**: Acquisition mode

This bit is set and cleared by software to select the acquisition mode.

0: Normal acquisition mode (acquisition starts as soon as START bit is set)
 1: Synchronized acquisition mode (acquisition starts if START bit is set and when the selected signal is detected on the SYNC input pin)

Note: This bit must not be modified when an acquisition is ongoing.

Bit 1 **START**: Start a new acquisition

This bit is set by software to start a new acquisition. It is cleared by hardware as soon as the acquisition is complete or by software to cancel the ongoing acquisition.

0: Acquisition not started
 1: Start a new acquisition

Bit 0 **TSCE**: Touch sensing controller enable

This bit is set and cleared by software to enable/disable the touch sensing controller.

0: Touch sensing controller disabled
 1: Touch sensing controller enabled

Note: When the touch sensing controller is disabled, TSC registers settings have no effect.

19.6.2 TSC interrupt enable register (TSC_IER)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEIE**: Max count error interrupt enable

This bit is set and cleared by software to enable/disable the max count error interrupt.

0: Max count error interrupt disabled

1: Max count error interrupt enabled

Bit 0 **EOAIE**: End of acquisition interrupt enable

This bit is set and cleared by software to enable/disable the end of acquisition interrupt.

0: End of acquisition interrupt disabled

1: End of acquisition interrupt enabled

19.6.3 TSC interrupt clear register (TSC_ICR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEIC**: Max count error interrupt clear

This bit is set by software to clear the max count error flag and it is cleared by hardware when the flag is reset. Writing a '0' has no effect.

0: No effect

1: Clears the corresponding MCEF of the TSC_ISR register

Bit 0 **EOAIC**: End of acquisition interrupt clear

This bit is set by software to clear the end of acquisition flag and it is cleared by hardware when the flag is reset. Writing a '0' has no effect.

0: No effect

1: Clears the corresponding EOAF of the TSC_ISR register

19.6.4 TSC interrupt status register (TSC_ISR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEF**: Max count error flag

This bit is set by hardware as soon as an analog I/O group counter reaches the max count value specified. It is cleared by software writing 1 to the bit MCEIC of the TSC_ICR register.

0: No max count error (MCE) detected

1: Max count error (MCE) detected

Bit 0 **EOAF**: End of acquisition flag

This bit is set by hardware when the acquisition of all enabled group is complete (all GxS bits of all enabled analog I/O groups are set or when a max count error is detected). It is cleared by software writing 1 to the bit EOAFIC of the TSC_ICR register.

0: Acquisition is ongoing or not started

1: Acquisition is complete

19.6.5 TSC I/O hysteresis control register (TSC_IOHCR)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
rw	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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Gx_IOy**: Gx_IOy Schmitt trigger hysteresis mode

These bits are set and cleared by software to enable/disable the Gx_IOy Schmitt trigger hysteresis.

0: Gx_IOy Schmitt trigger hysteresis disabled

1: Gx_IOy Schmitt trigger hysteresis enabled

Note: These bits control the I/O Schmitt trigger hysteresis whatever the I/O control mode is (even if controlled by standard GPIO registers).

19.6.6 TSC I/O analog switch control register (TSC_IOASCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Gx_IOy**: Gx_IOy analog switch enable

These bits are set and cleared by software to enable/disable the Gx_IOy analog switch.

0: Gx_IOy analog switch disabled (opened)

1: Gx_IOy analog switch enabled (closed)

Note: These bits control the I/O analog switch whatever the I/O control mode is (even if controlled by standard GPIO registers).

19.6.7 TSC I/O sampling control register (TSC_IOSCR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Gx_IOy**: Gx_IOy sampling mode

These bits are set and cleared by software to configure the Gx_IOy as a sampling capacitor I/O. Only one I/O per analog I/O group must be defined as sampling capacitor.

0: Gx_IOy unused

1: Gx_IOy used as sampling capacitor

Note: These bits must not be modified when an acquisition is ongoing.

During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

19.6.8 TSC I/O channel control register (TSC_IOCCR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Gx_IOy**: Gx_IOy channel mode

These bits are set and cleared by software to configure the Gx_IOy as a channel I/O.

0: Gx_IOy unused

1: Gx_IOy used as channel

Note: These bits must not be modified when an acquisition is ongoing.

During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOCRR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

19.6.9 TSC I/O group control status register (TSC_ILOGCSR)

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.	G8S	G7S	G6S	G5S	G4S	G3S	G2S	G1S
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G8E	G7E	G6E	G5E	G4E	G3E	G2E	G1E
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **GxS**: Analog I/O group x status

These bits are set by hardware when the acquisition on the corresponding enabled analog I/O group x is complete. They are cleared by hardware when a new acquisition is started.

0: Acquisition on analog I/O group x is ongoing or not started

1: Acquisition on analog I/O group x is complete

Note: When a max count error is detected the remaining GxS bits of the enabled analog I/O groups are not set.

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

Bits 7:0 **GxE**: Analog I/O group x enable

These bits are set and cleared by software to enable/disable the acquisition (counter is counting) on the corresponding analog I/O group x.

0: Acquisition on analog I/O group x disabled

1: Acquisition on analog I/O group x enabled

19.6.10 TSC I/O group x counter register (TSC_ILOGxCR) (x = 1..8)

Address offset: 0x30 + 0x04 x Analog I/O group number

[illegible]

Bits 13:0 **CNT[13:0]**: Counter value

19.6.11 TSC register map

[illegible]

Table 112. TSC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0024	Reserved																																
0x0028	TSC_IOCRR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	Reserved																																
0x0030	TSC_IQGCSR	Res	Res	Res	Res	Res	Res	Res	Res	G8S	G7S	G6S	G5S	G4S	G3S	G2S	G1S	Res	Res	Res	Res	Res	Res	Res	Res	G8E	G7E	G6E	G5E	G4E	G3E	G2E	G1E
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x0034	TSC_IQG1CR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0038	TSC_IQG2CR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x003C	TSC_IQG3CR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0040	TSC_IQG4CR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	TSC_IQG5CR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	TSC_IQG6CR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	TSC_IQG7CR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	TSC_IQG8CR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

20 Advanced-control timers (TIM1)

20.1 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.

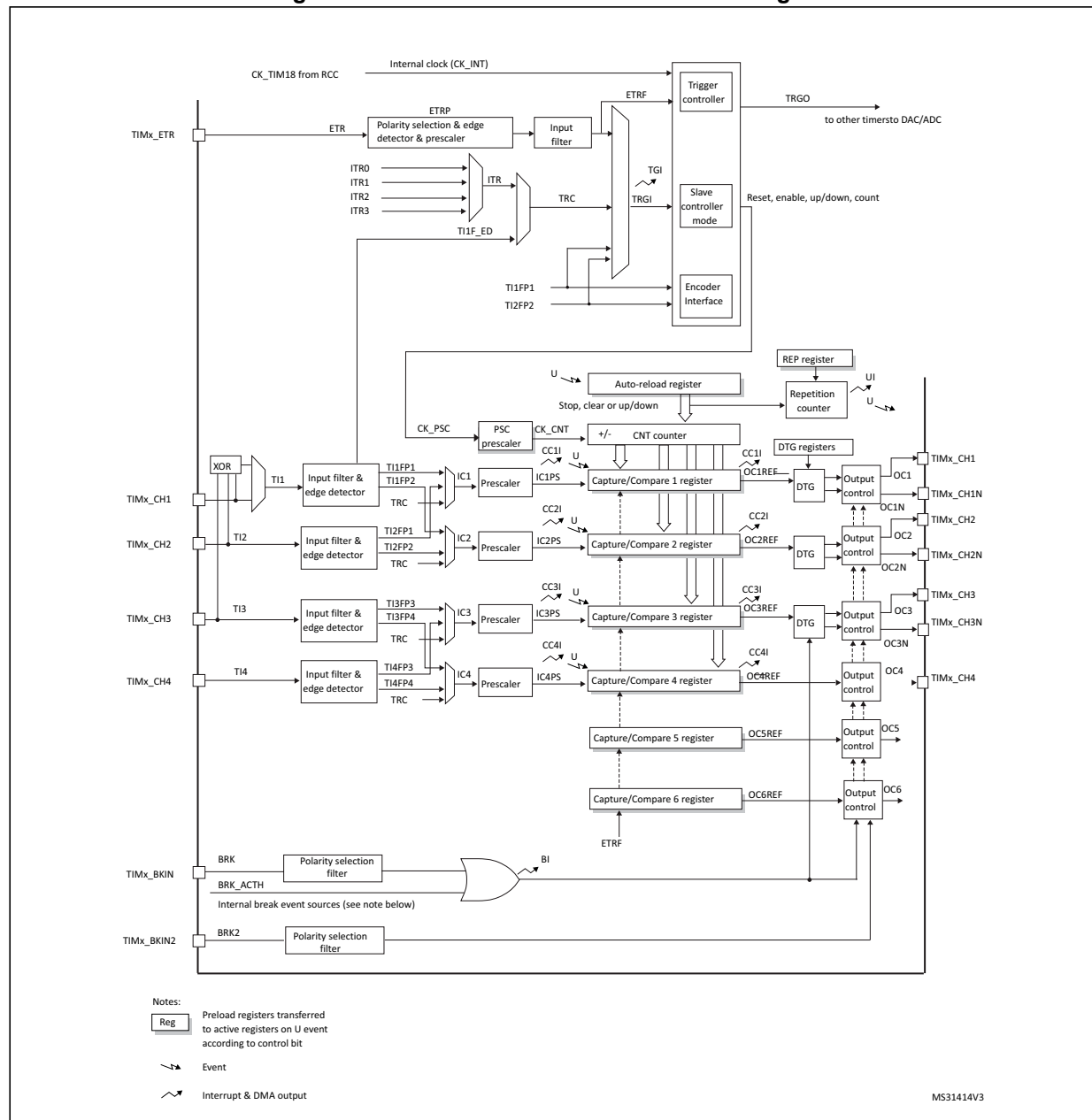
The advanced-control (TIM1) and general-purpose (TIMx) timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 20.3.25](#).

20.2 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 132. 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 10.2.7: Clock security system \(CSS\)](#)
 - A PVD output
 - SRAM parity error signal
 - Cortex-M4 LOCKUP (Hardfault) output.
 - COMPx output, x = 1, 2 and 6.

20.3 TIM1 functional description

20.3.1 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 133 and *Figure 134* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 133. Counter timing diagram with prescaler division change from 1 to 2

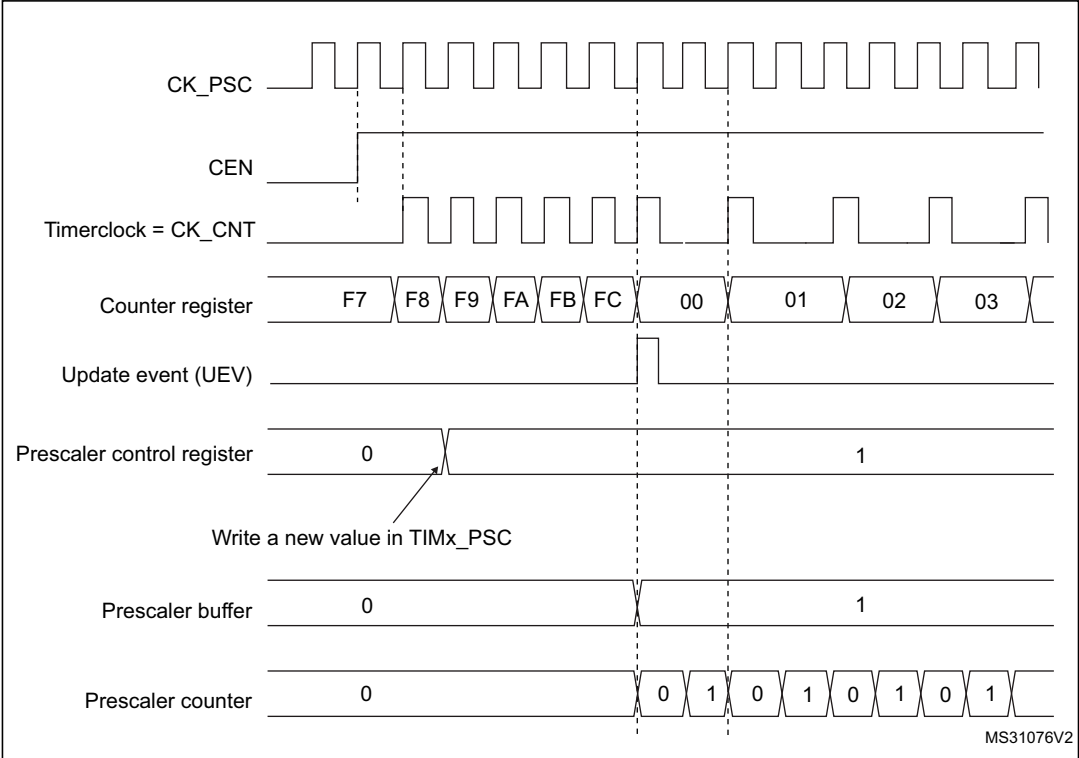
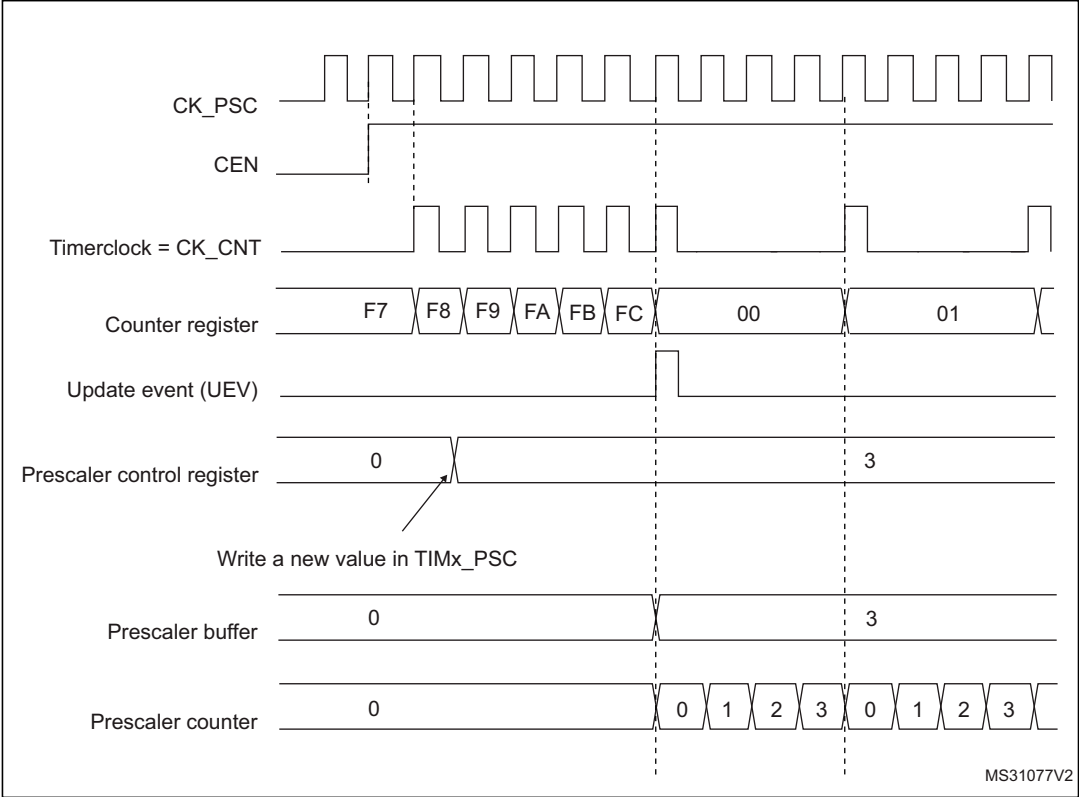


Figure 134. Counter timing diagram with prescaler division change from 1 to 4



20.3.2 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 135. Counter timing diagram, internal clock divided by 1

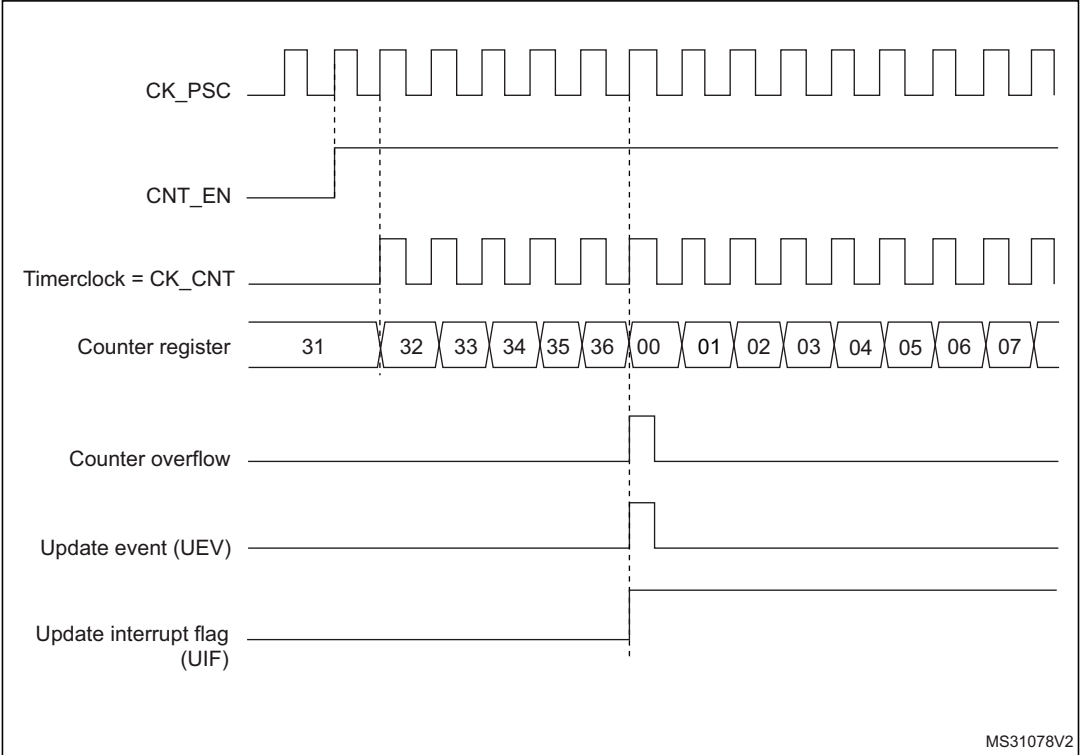


Figure 136. Counter timing diagram, internal clock divided by 2

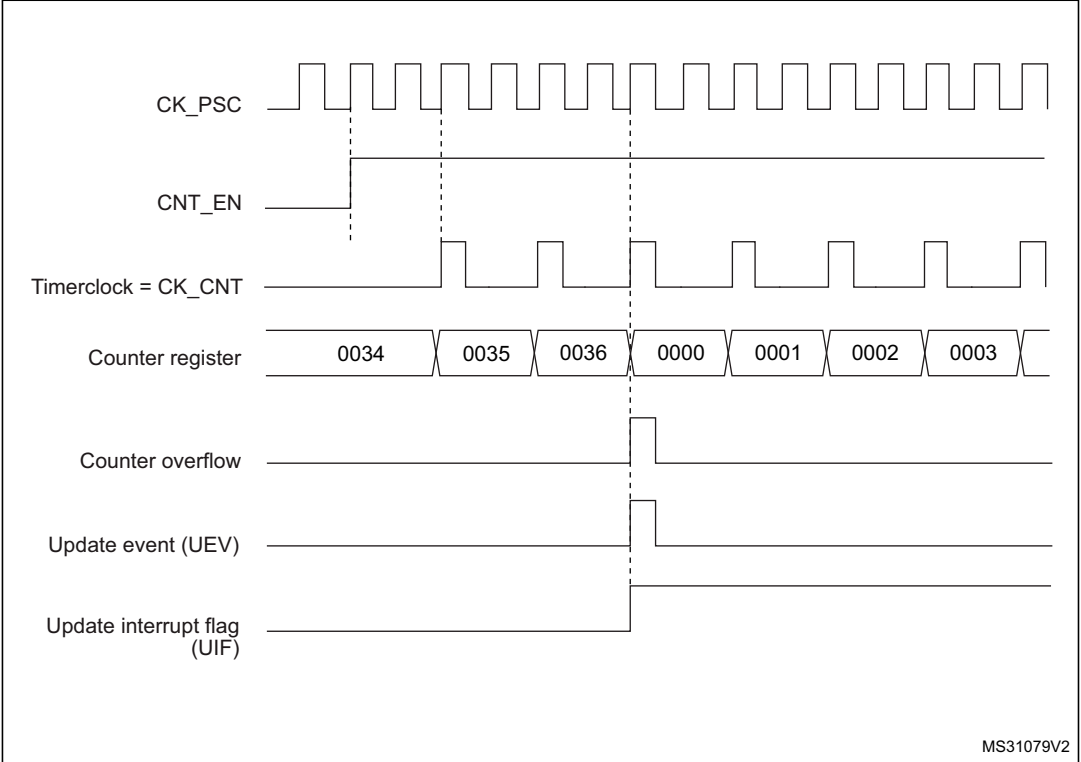


Figure 137. Counter timing diagram, internal clock divided by 4

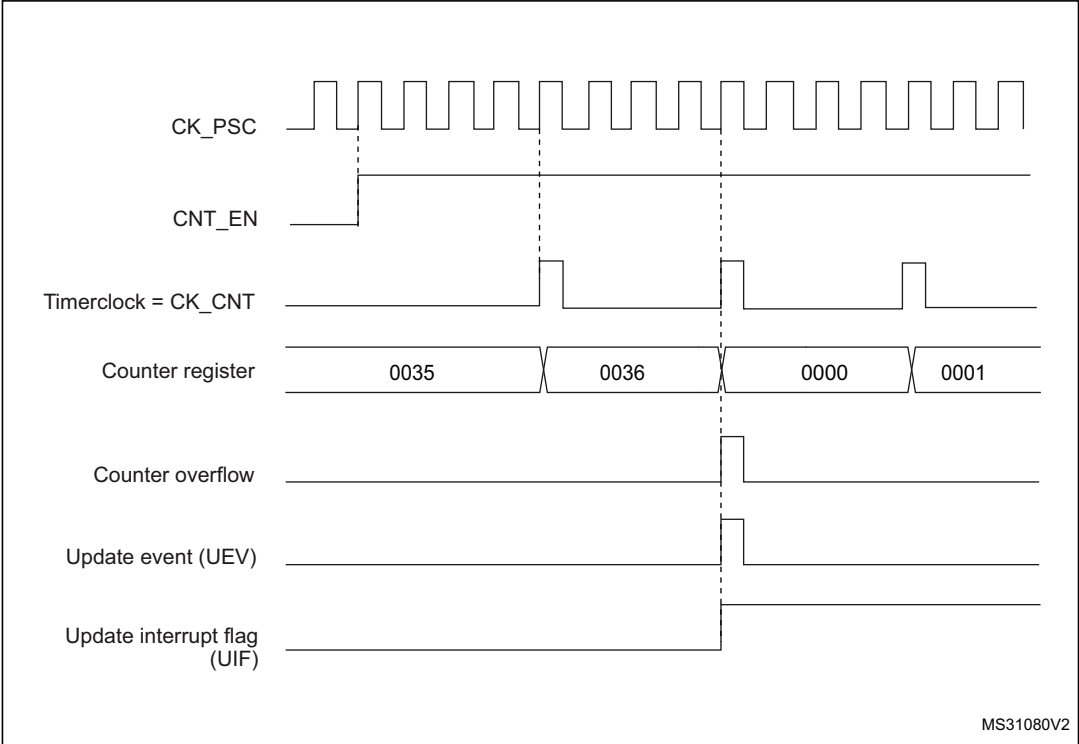


Figure 138. Counter timing diagram, internal clock divided by N

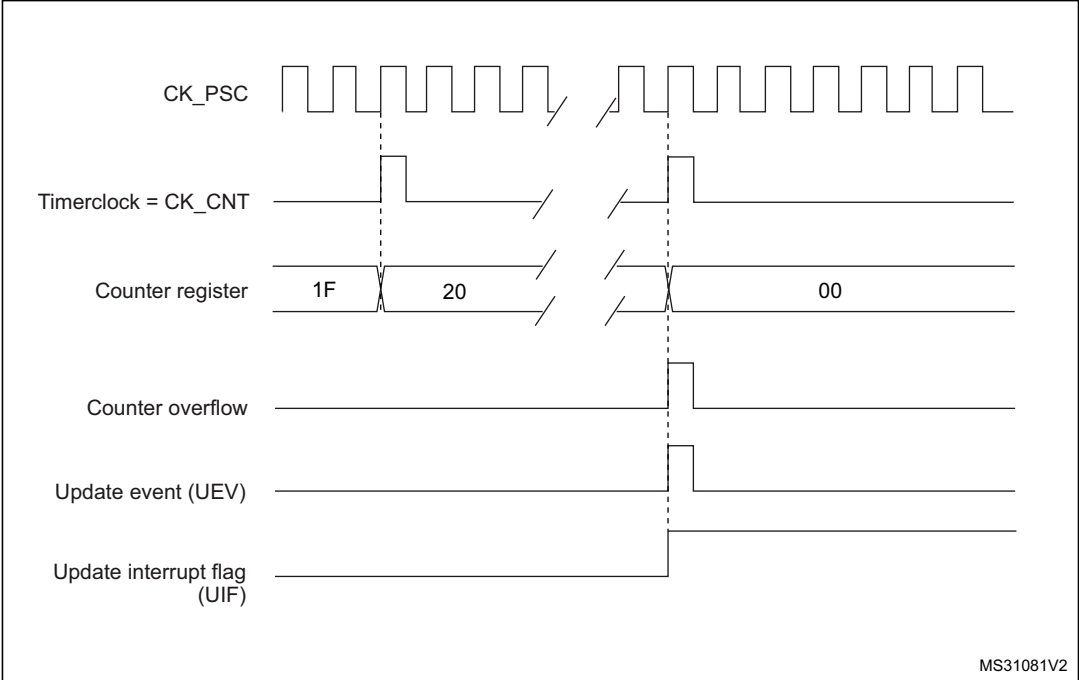


Figure 139. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

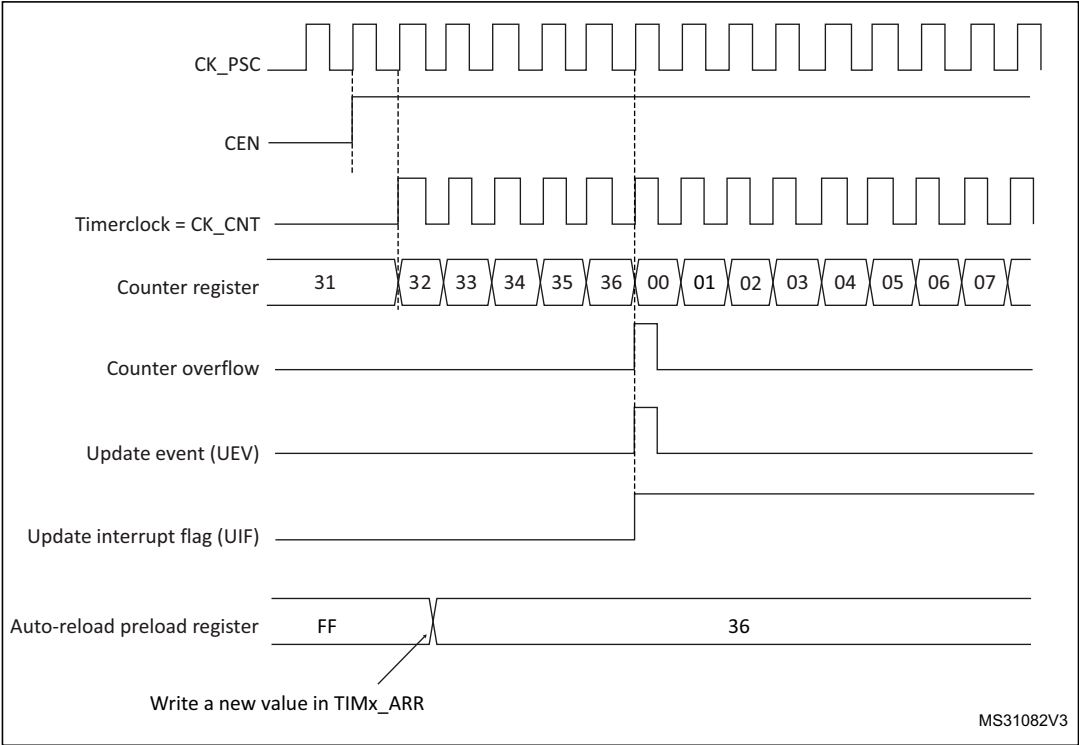
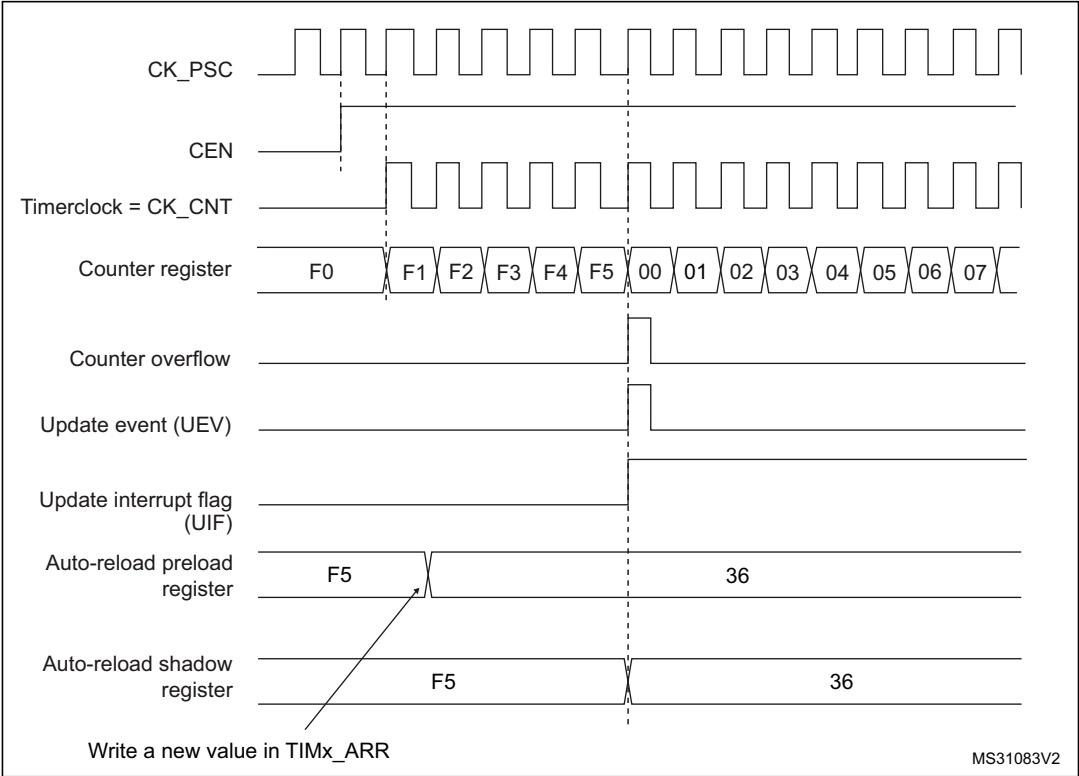


Figure 140. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



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 141. Counter timing diagram, internal clock divided by 1

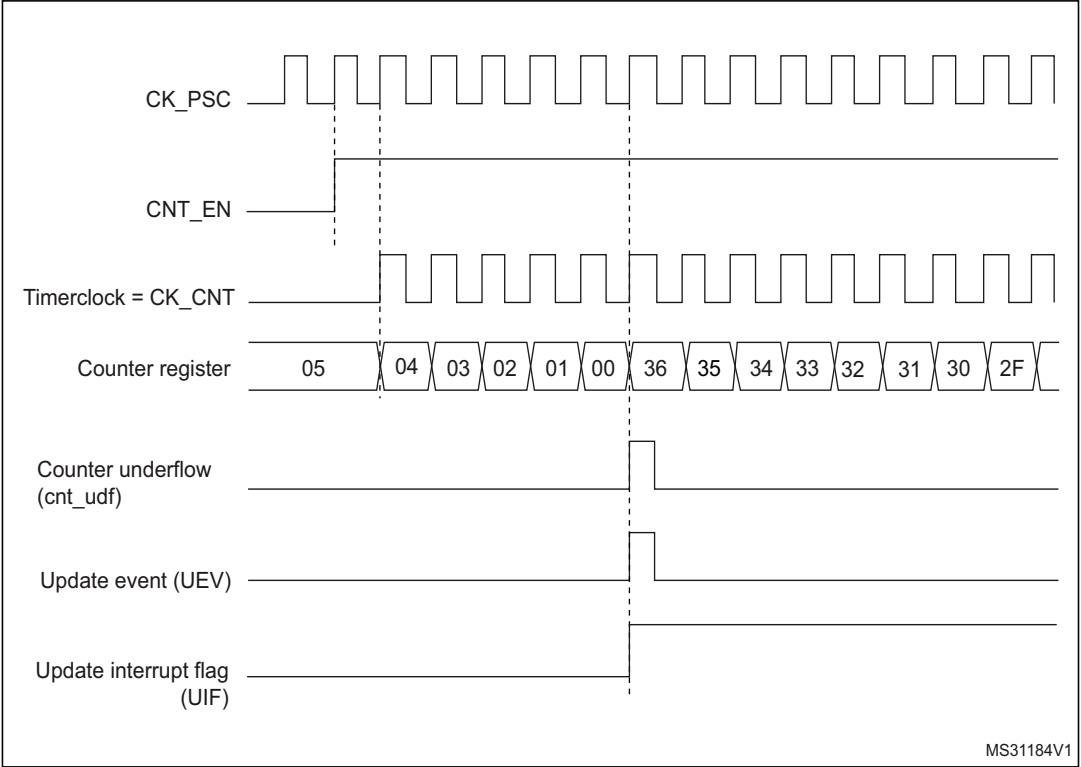


Figure 142. Counter timing diagram, internal clock divided by 2

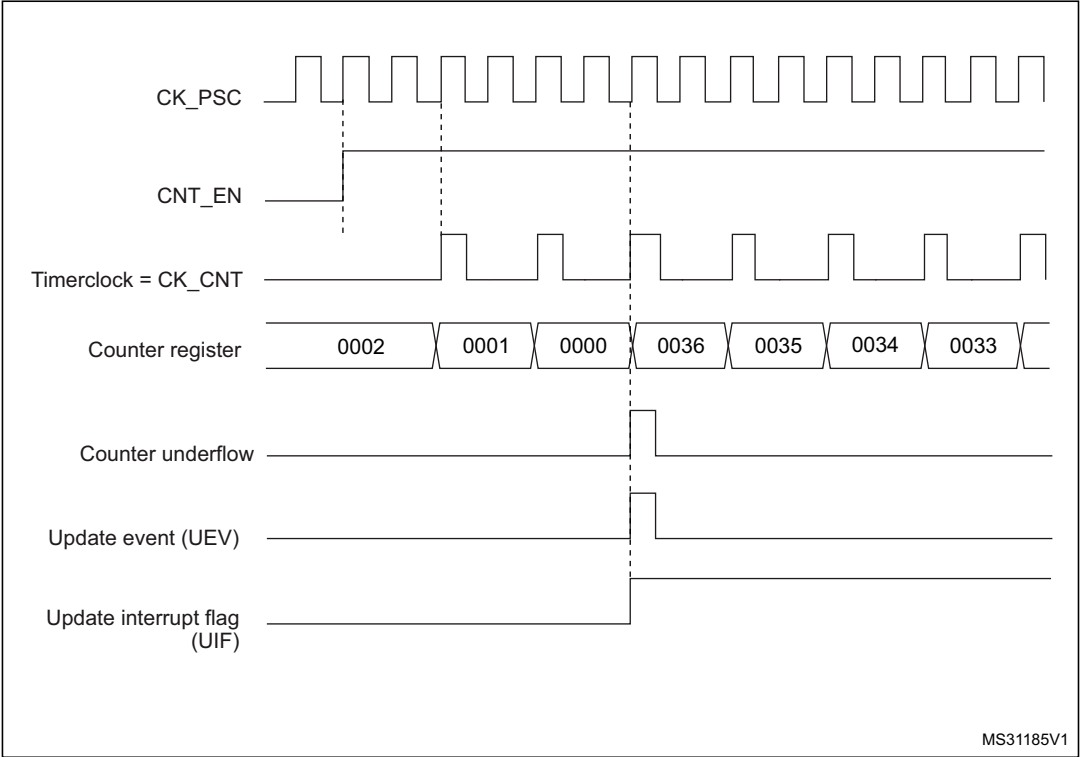


Figure 143. Counter timing diagram, internal clock divided by 4

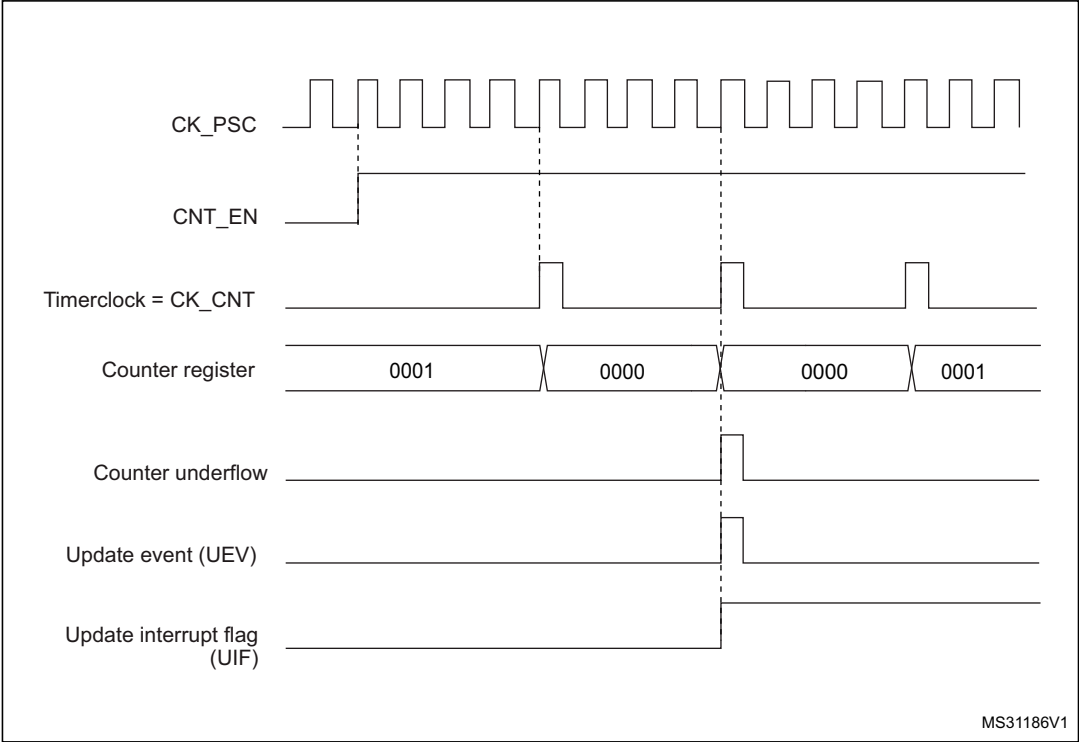


Figure 144. Counter timing diagram, internal clock divided by N

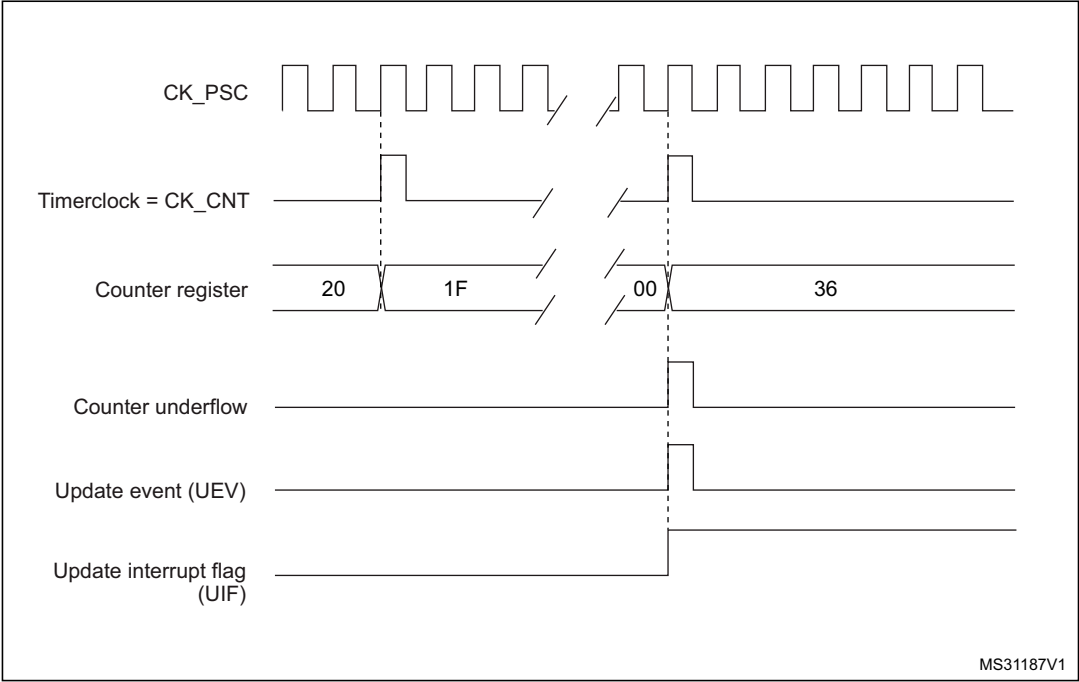
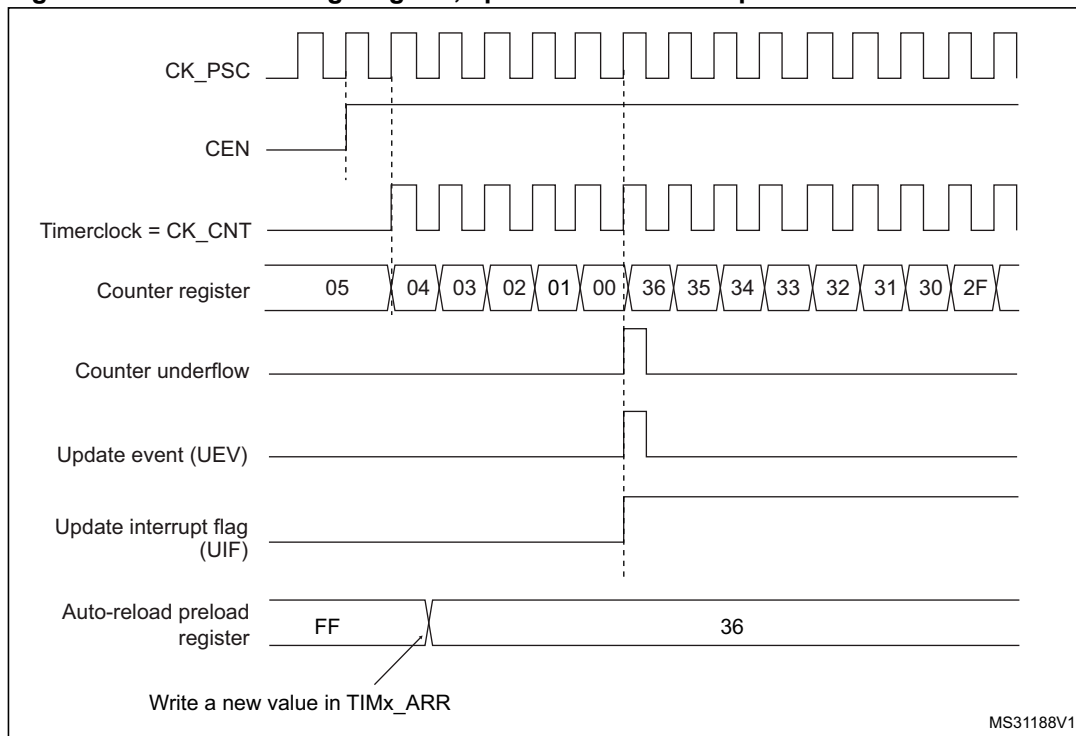


Figure 145. Counter timing diagram, update event when repetition counter is not used

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

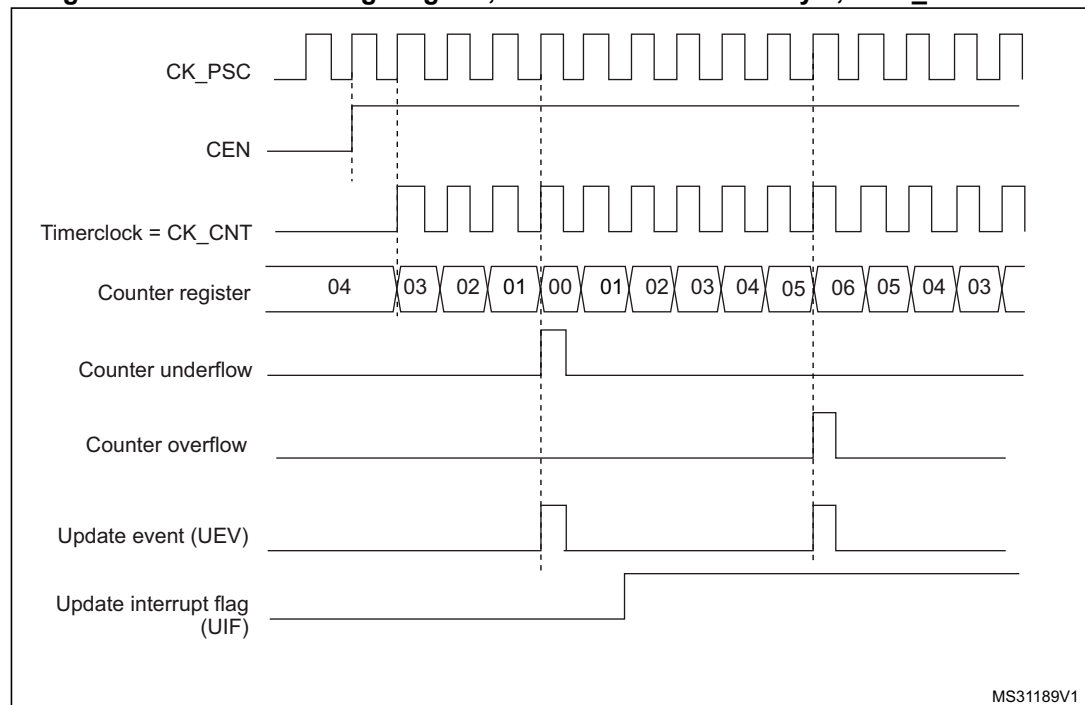
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

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 146. 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 20.4: TIM1 registers](#)).

Figure 147. Counter timing diagram, internal clock divided by 2

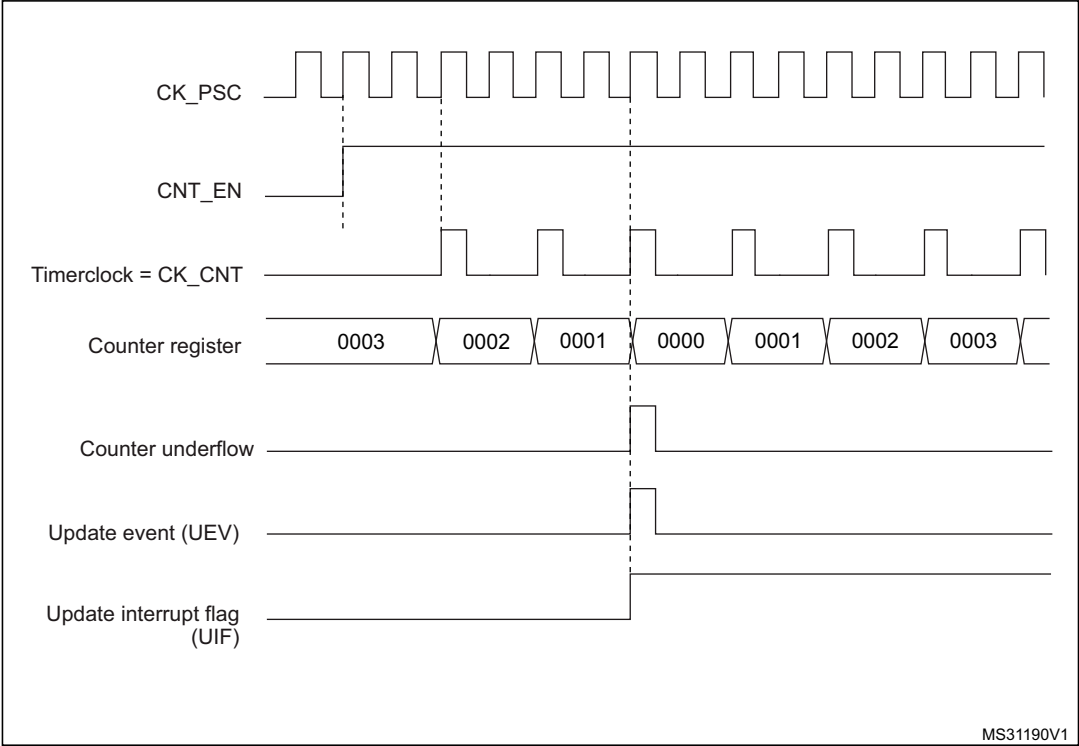


Figure 148. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

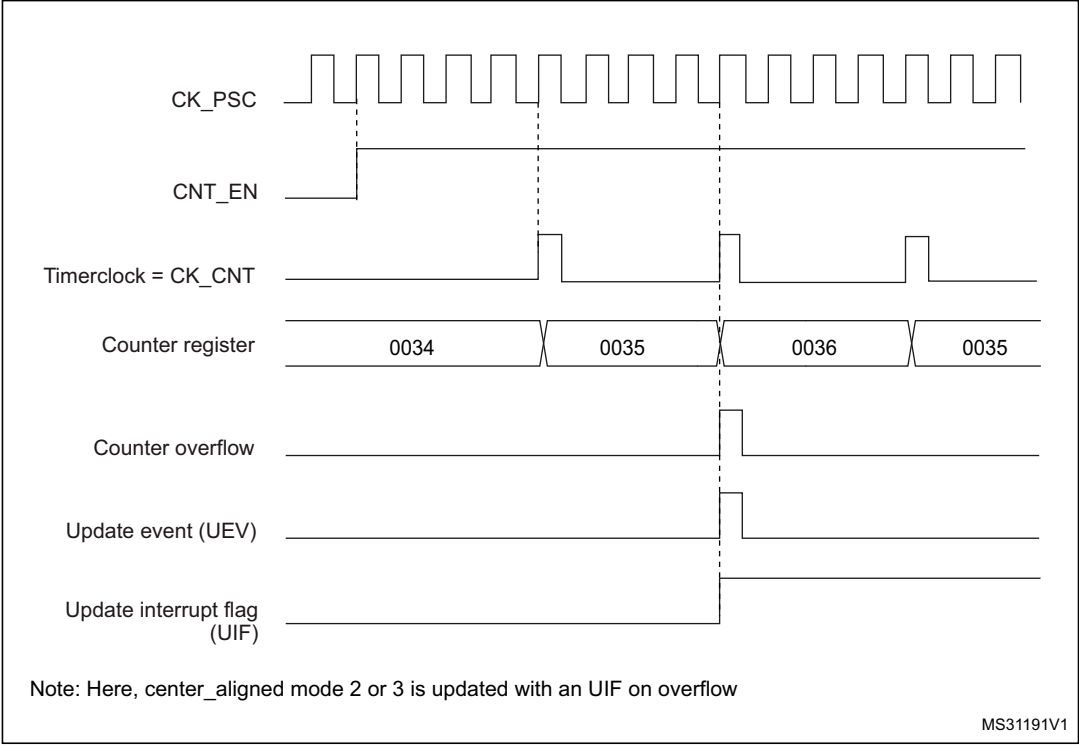


Figure 149. Counter timing diagram, internal clock divided by N

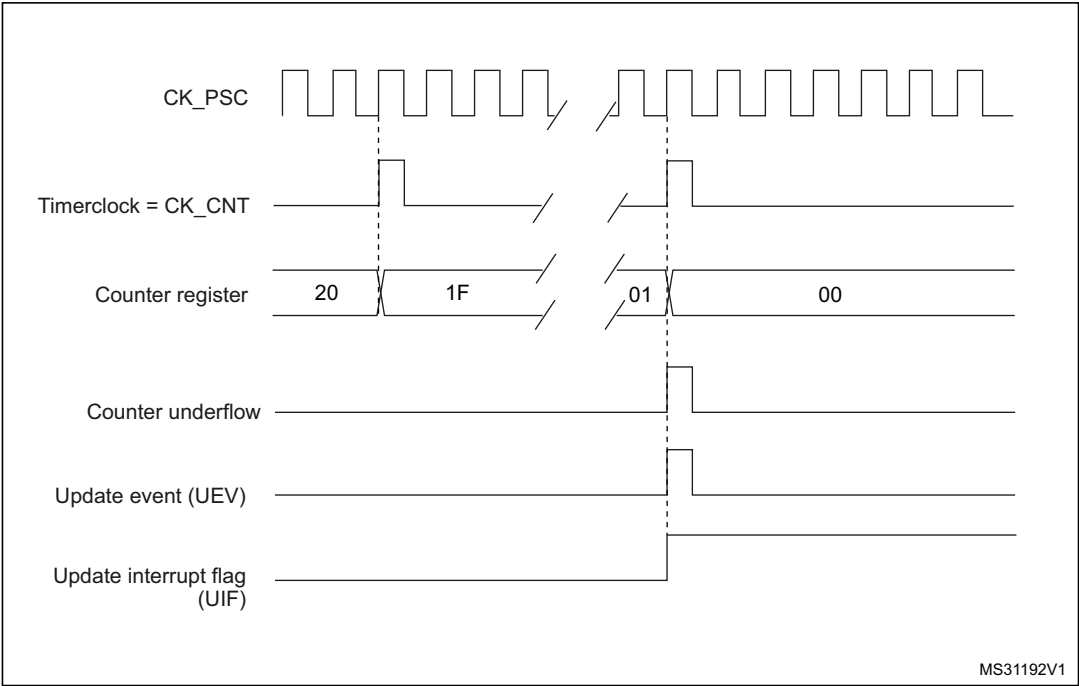


Figure 150. Counter timing diagram, update event with ARPE=1 (counter underflow)

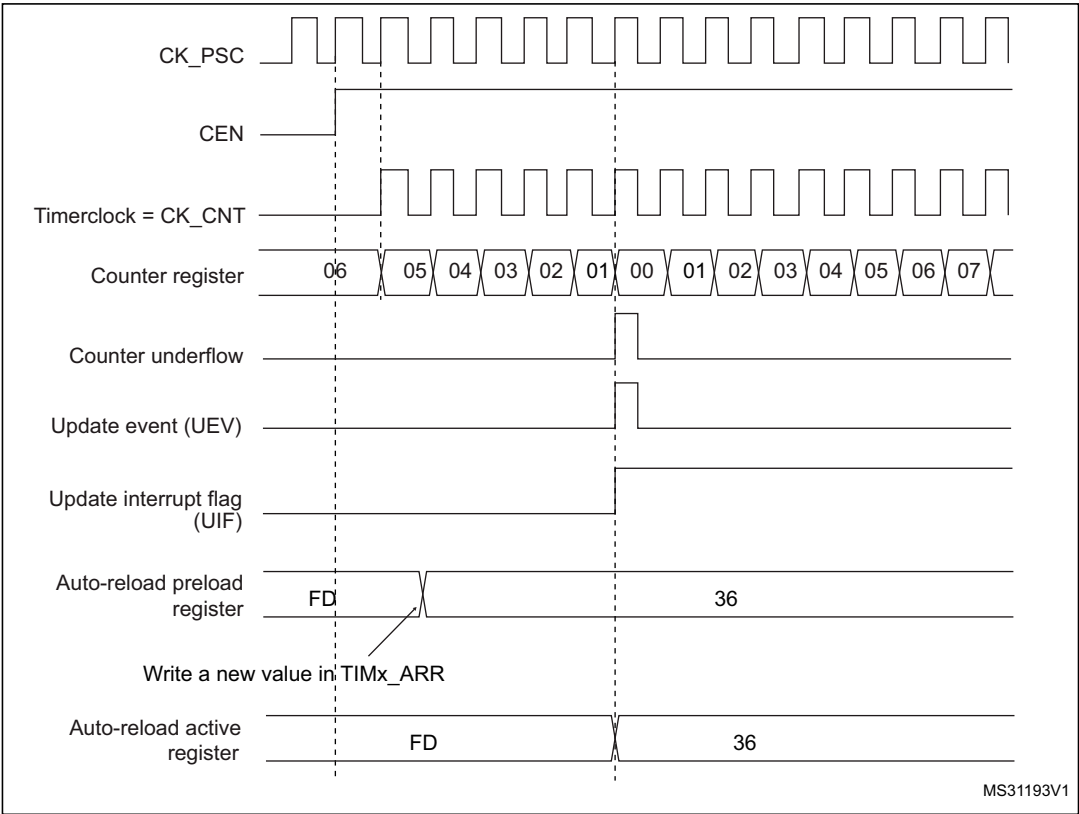
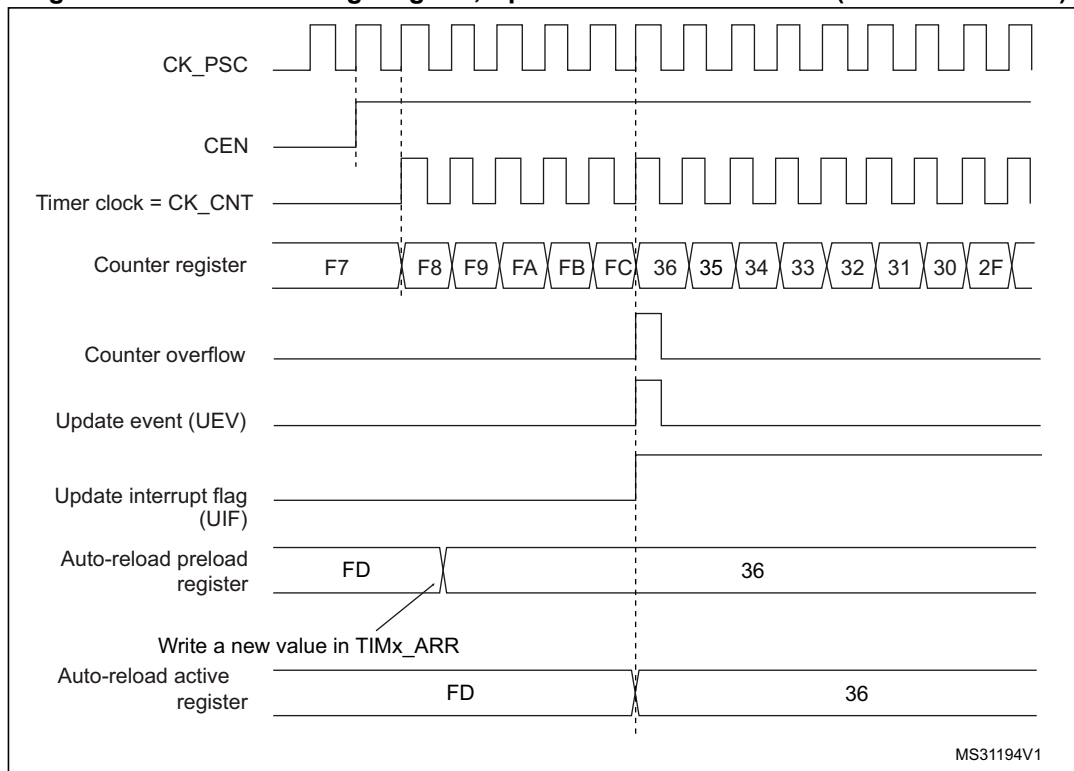


Figure 151. Counter timing diagram, Update event with ARPE=1 (counter overflow)

20.3.3 Repetition counter

[Section 20.3.1: 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:

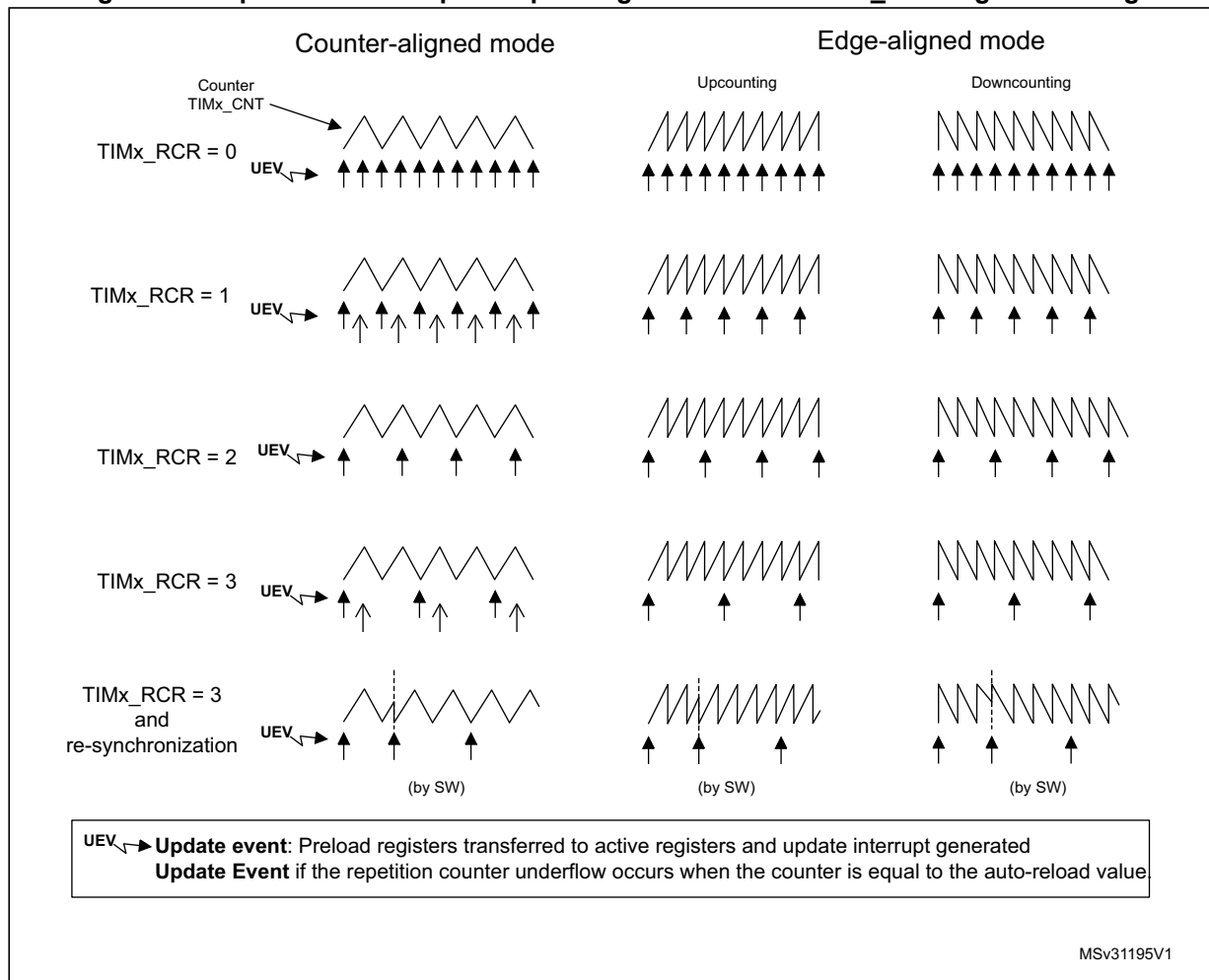
- At each counter overflow in upcounting mode,
 - At each counter underflow in downcounting mode,
 - At each counter overflow and at each counter underflow in center-aligned mode.
- Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle 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 152](#)). 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 overflow. If the RCR was written after launching the counter, the UEV occurs on the underflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 152. Update rate examples depending on mode and TIMx_RCR register settings



20.3.4 Clock selection

The counter clock can be provided by the following clock sources:

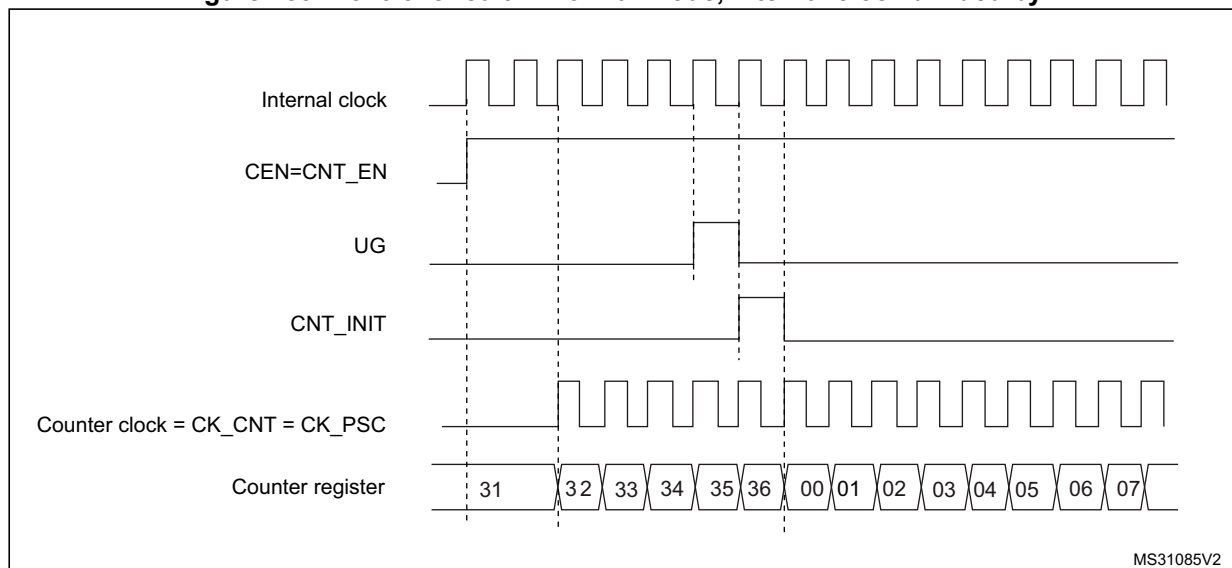
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

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 153 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

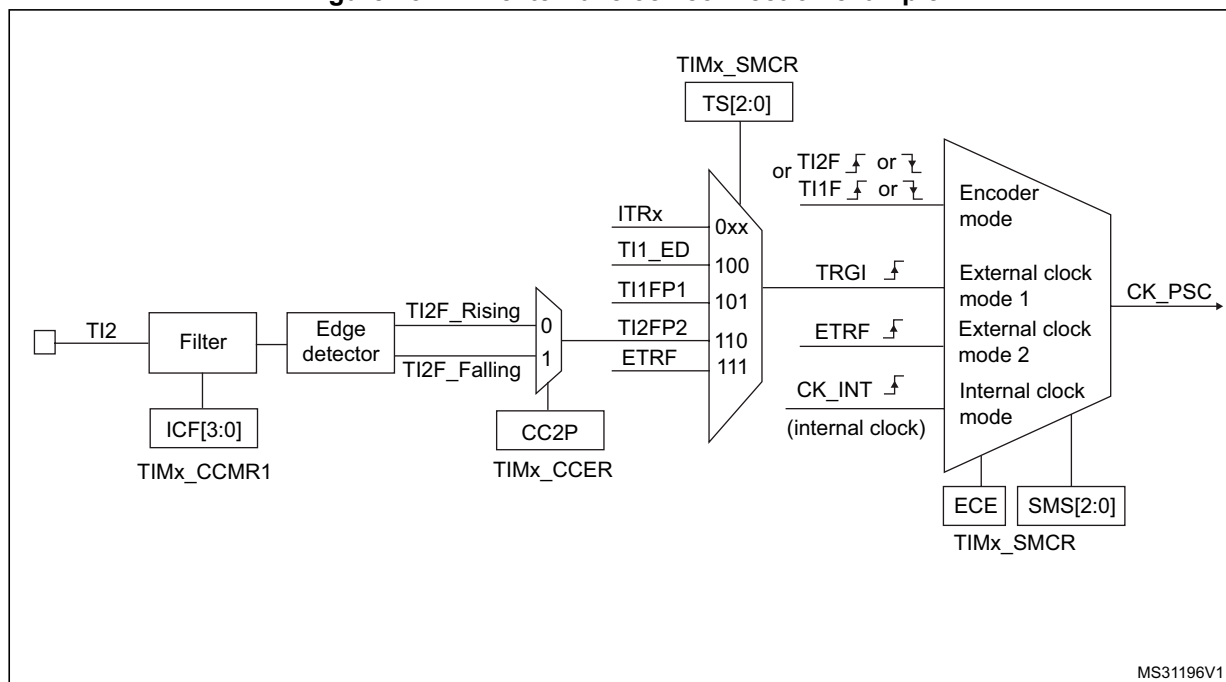
Figure 153. Control circuit in normal mode, internal clock divided by 1



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 154. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

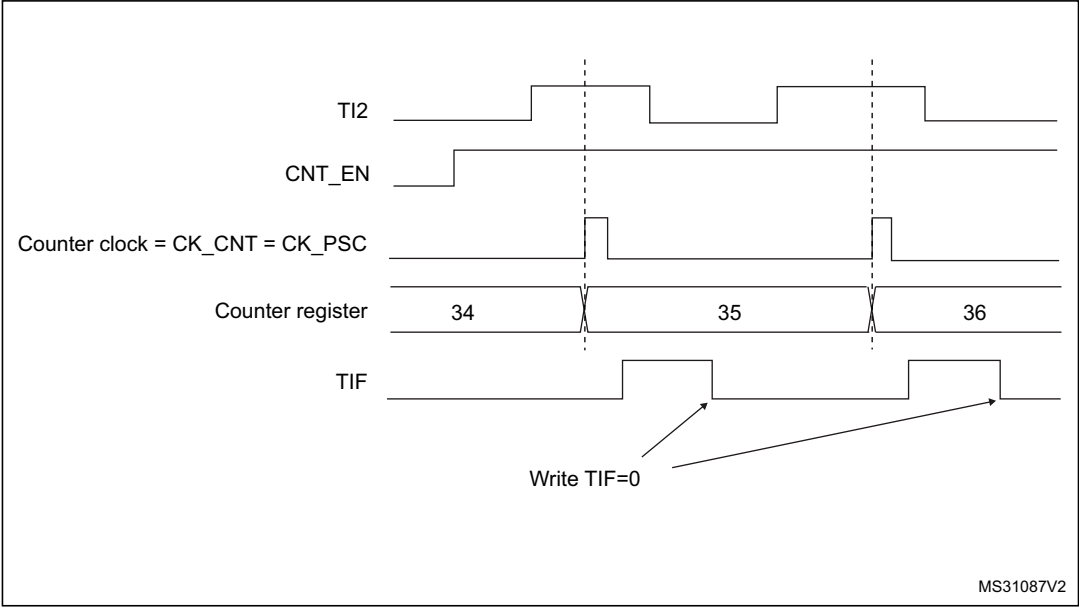
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the trigger input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so you don't need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 155. Control circuit in external clock mode 1

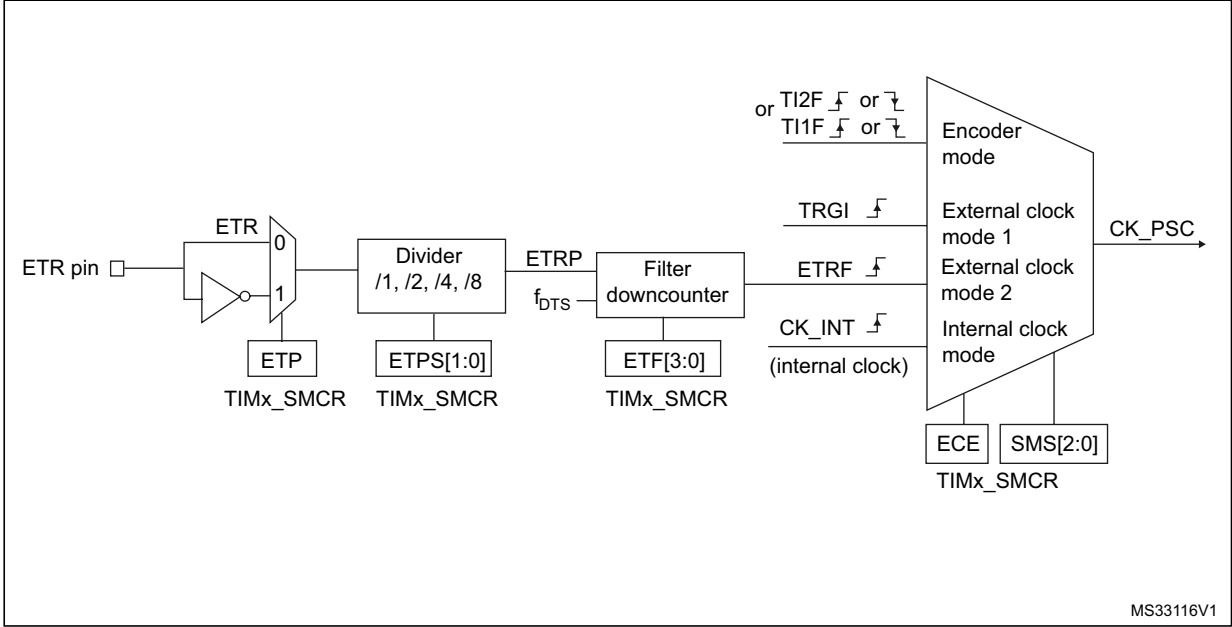


MS31087V2

External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register. The counter can count at each rising or falling edge on the external trigger input ETR. The [Figure 156](#) gives an overview of the external trigger input block.

Figure 156. External trigger input block



MS33116V1

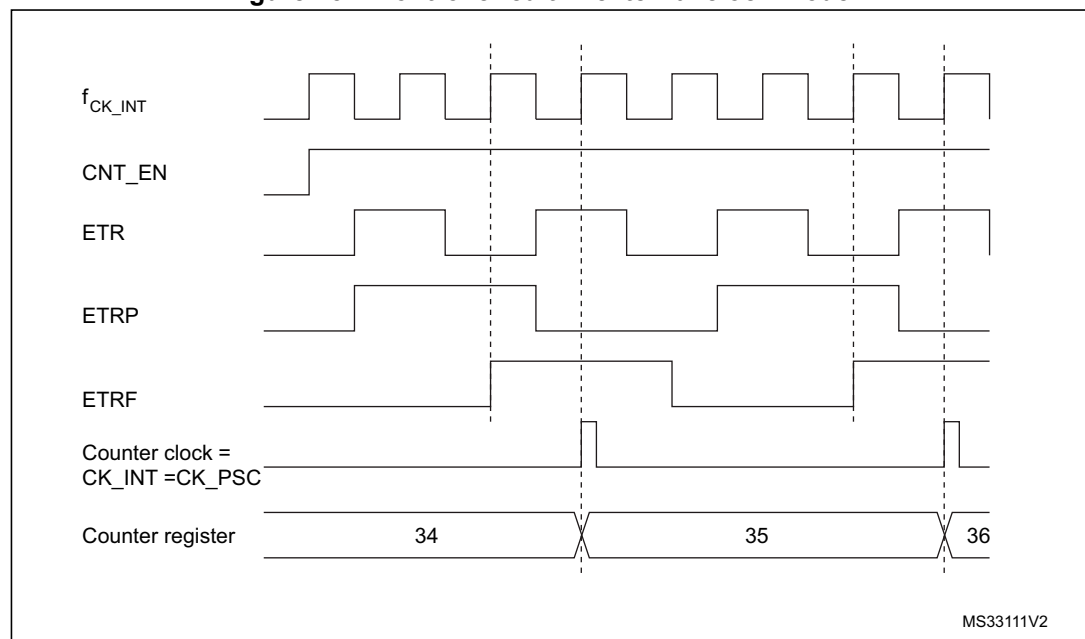
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write $ETF[3:0]=0000$ in the TIMx_SMCR register.
2. Set the prescaler by writing $ETPS[1:0]=01$ in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing $ETP=0$ in the TIMx_SMCR register
4. Enable external clock mode 2 by writing $ECE=1$ in the TIMx_SMCR register.
5. Enable the counter by writing $CEN=1$ in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 157. Control circuit in external clock mode 2



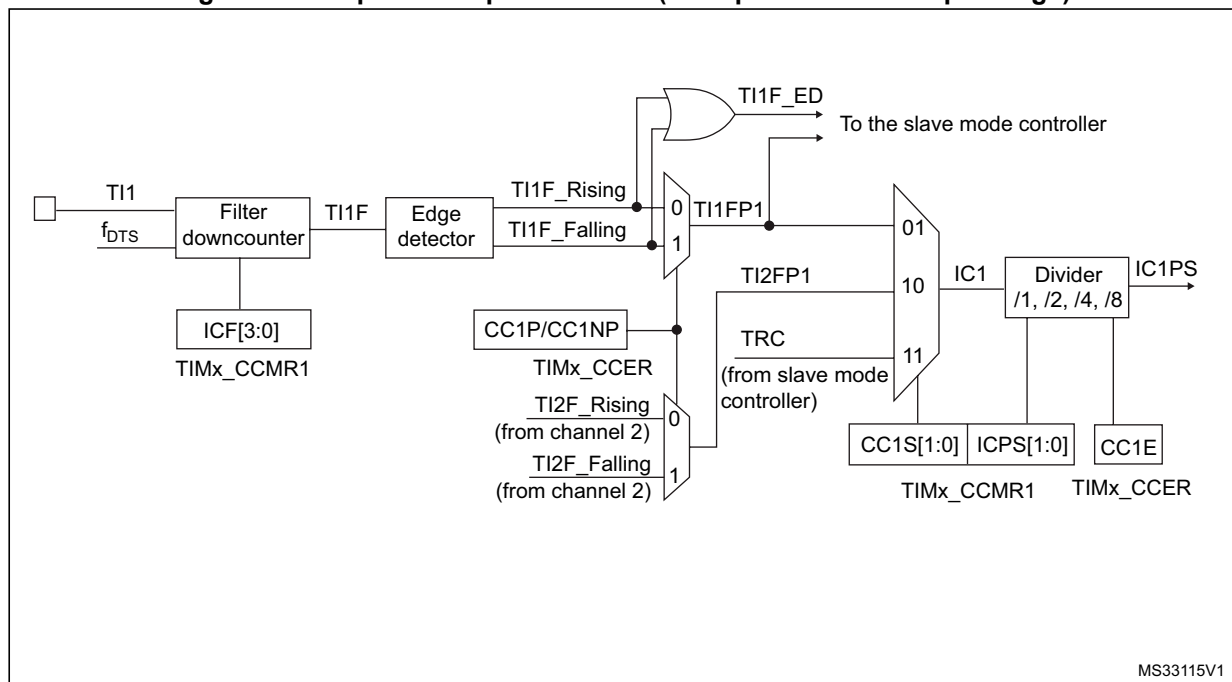
20.3.5 Capture/compare channels

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 158 to Figure 161 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 158. 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 159. Capture/compare channel 1 main circuit

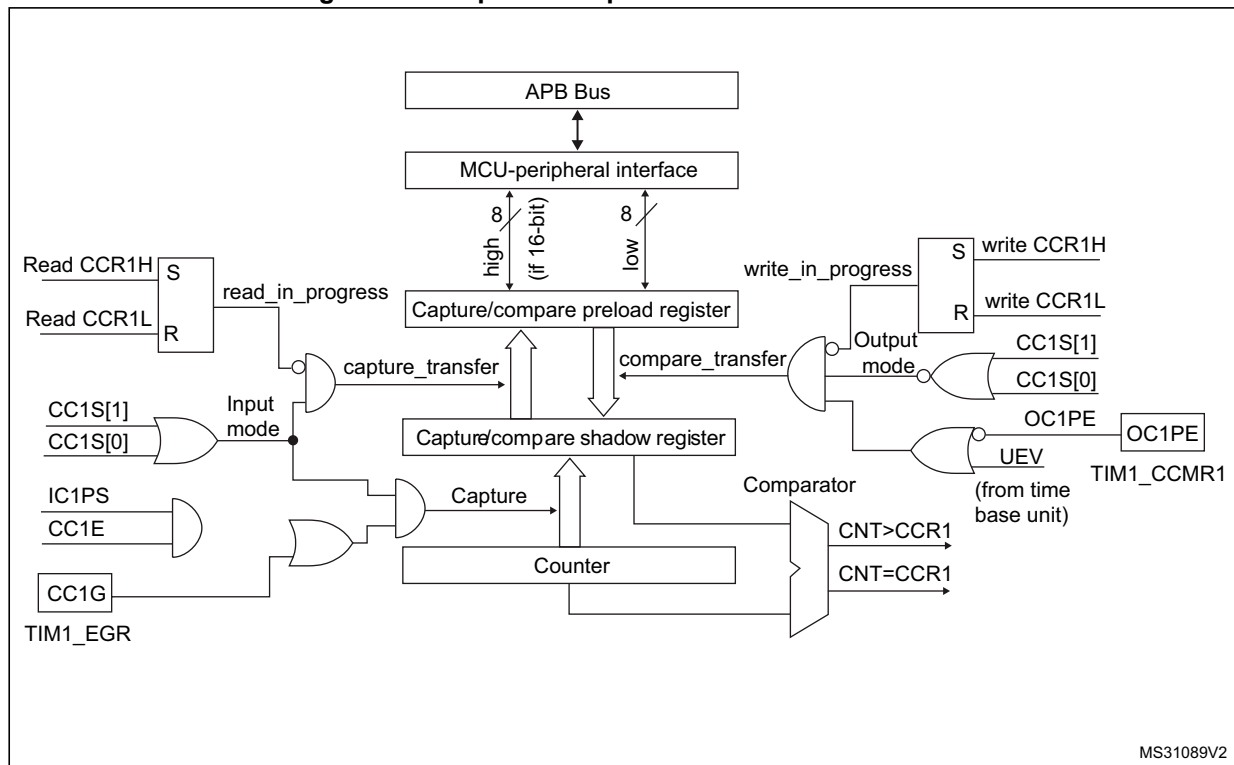
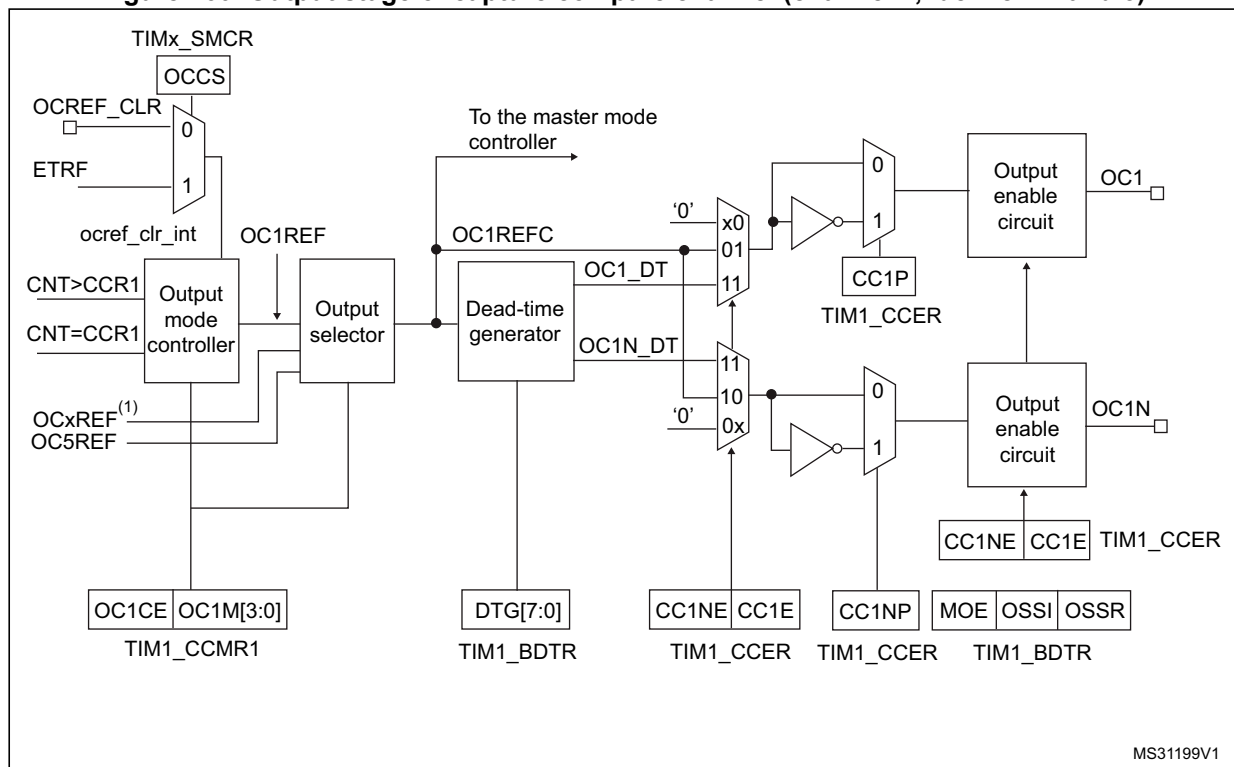
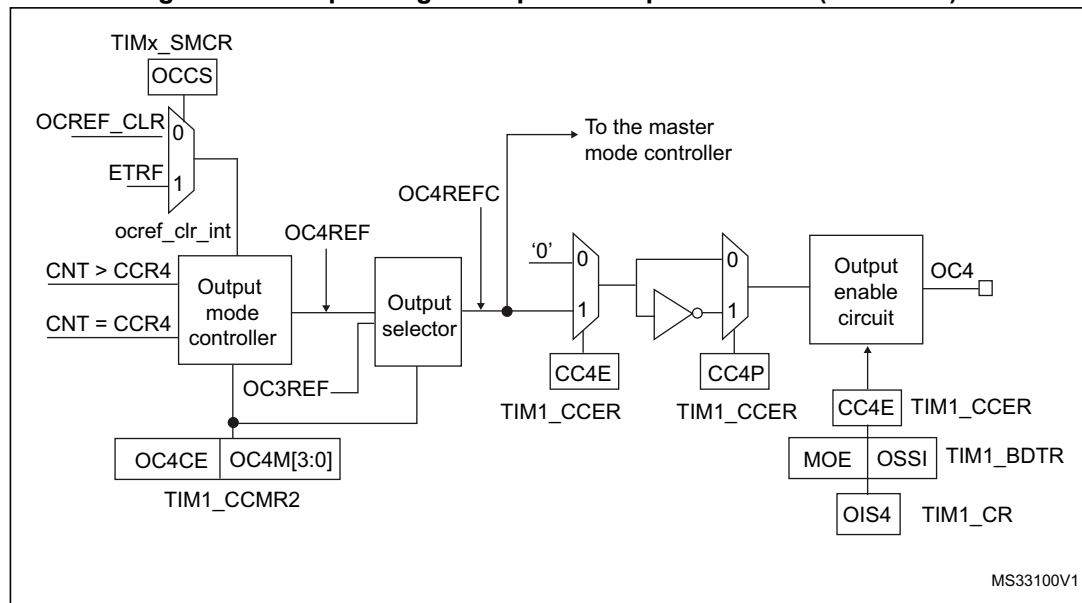
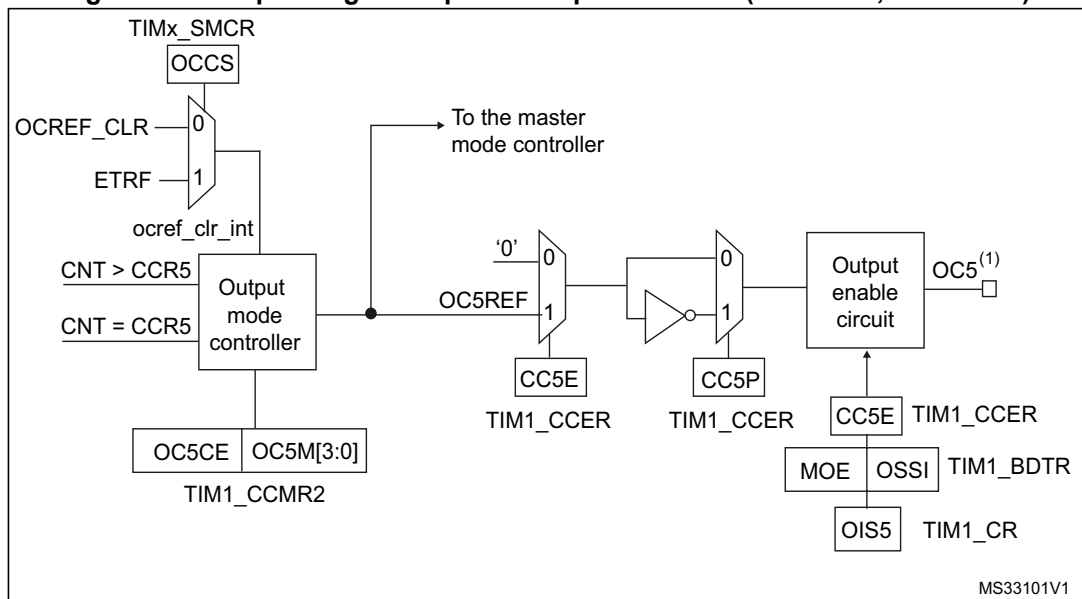


Figure 160. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)



1. OCxREF, where x is the rank of the complementary channel

Figure 161. Output stage of capture/compare channel (channel 4)**Figure 162. 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.

20.3.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCXIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCXIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at 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.
- 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).

- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

20.3.7 PWM input mode

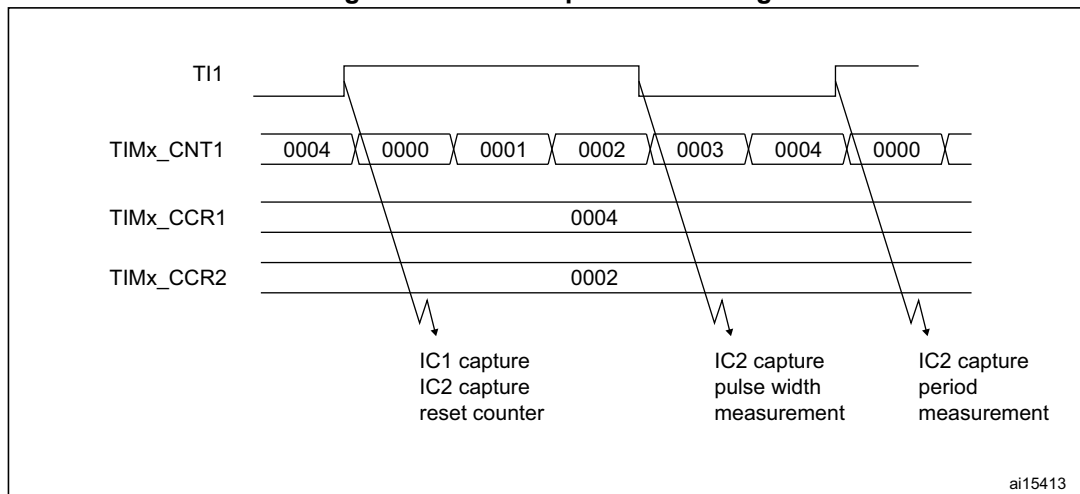
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP='10' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 163. PWM input mode timing



20.3.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCxREF/OCx) to its active level, you just need to write 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.

20.3.9 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 microcontroller (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.

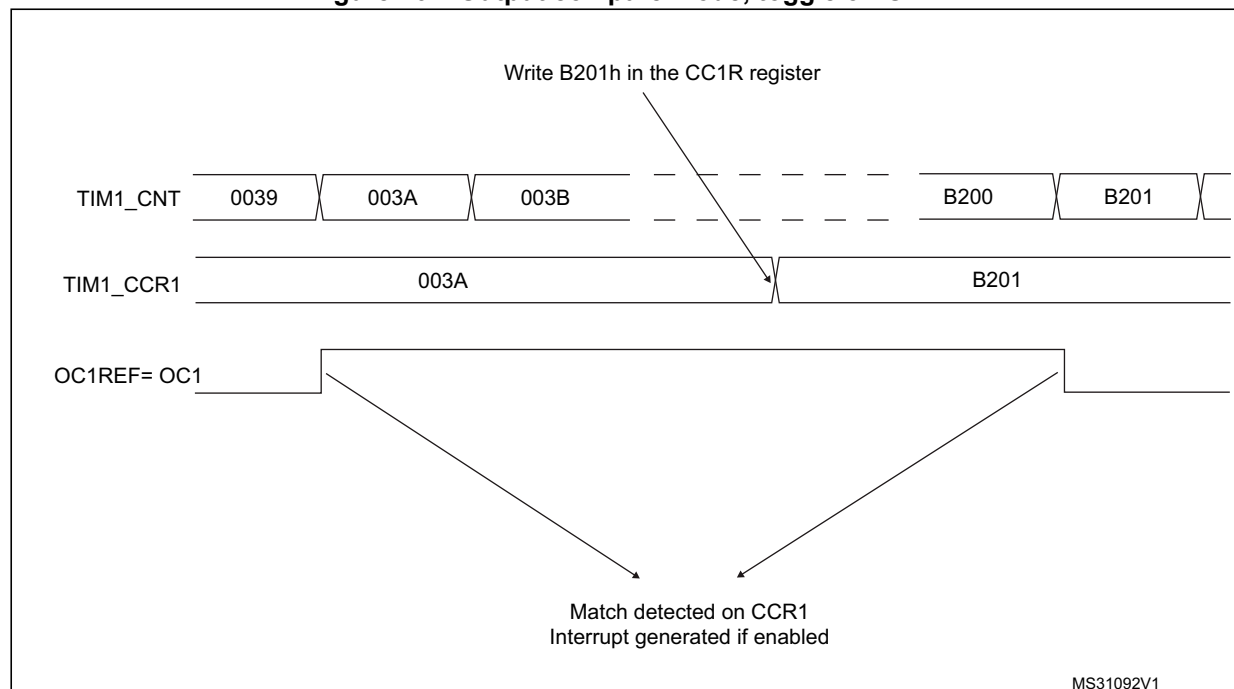
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

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 164](#).

Figure 164. Output compare mode, toggle on OC1



20.3.10 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '0110' (PWM mode 1) or '0111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether $TIMx_CCRx \leq TIMx_CNT$ or $TIMx_CNT \leq TIMx_CCRx$ (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

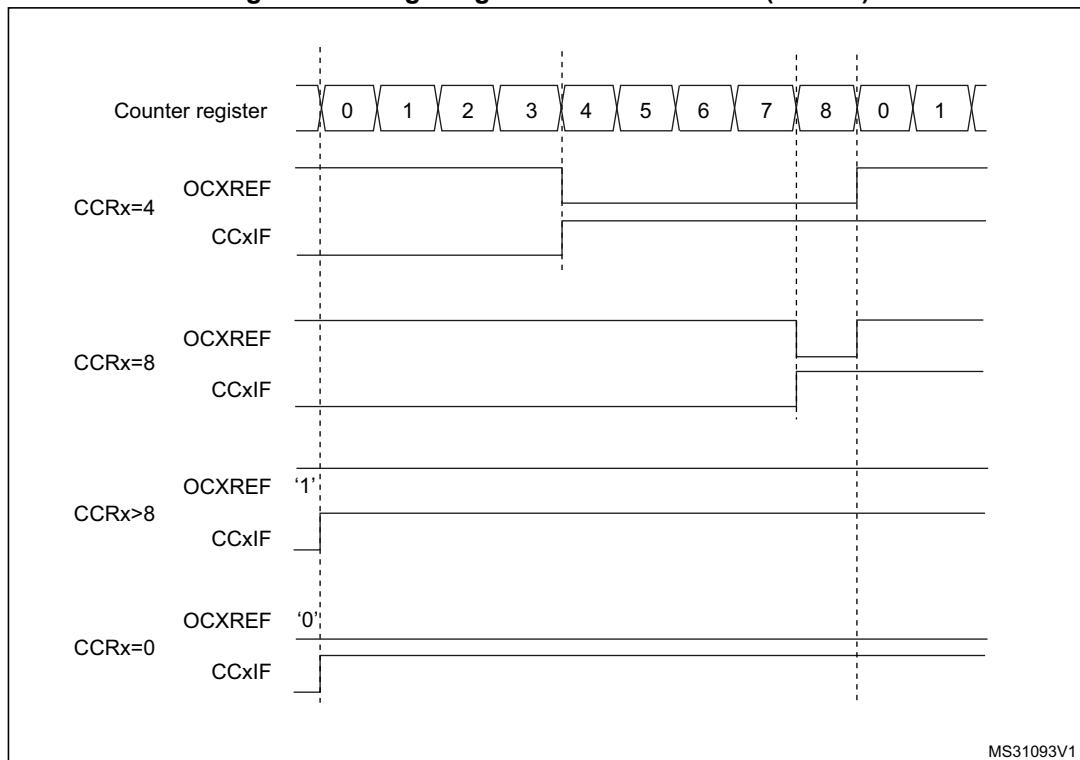
- Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the [Upcounting mode on page 457](#).

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 165](#) shows some edge-aligned PWM waveforms in an example where $TIMx_ARR=8$.

Figure 165. Edge-aligned PWM waveforms (ARR=8)



MS31093V1

- Downcounting configuration
Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the [Downcounting mode on page 461](#)
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.

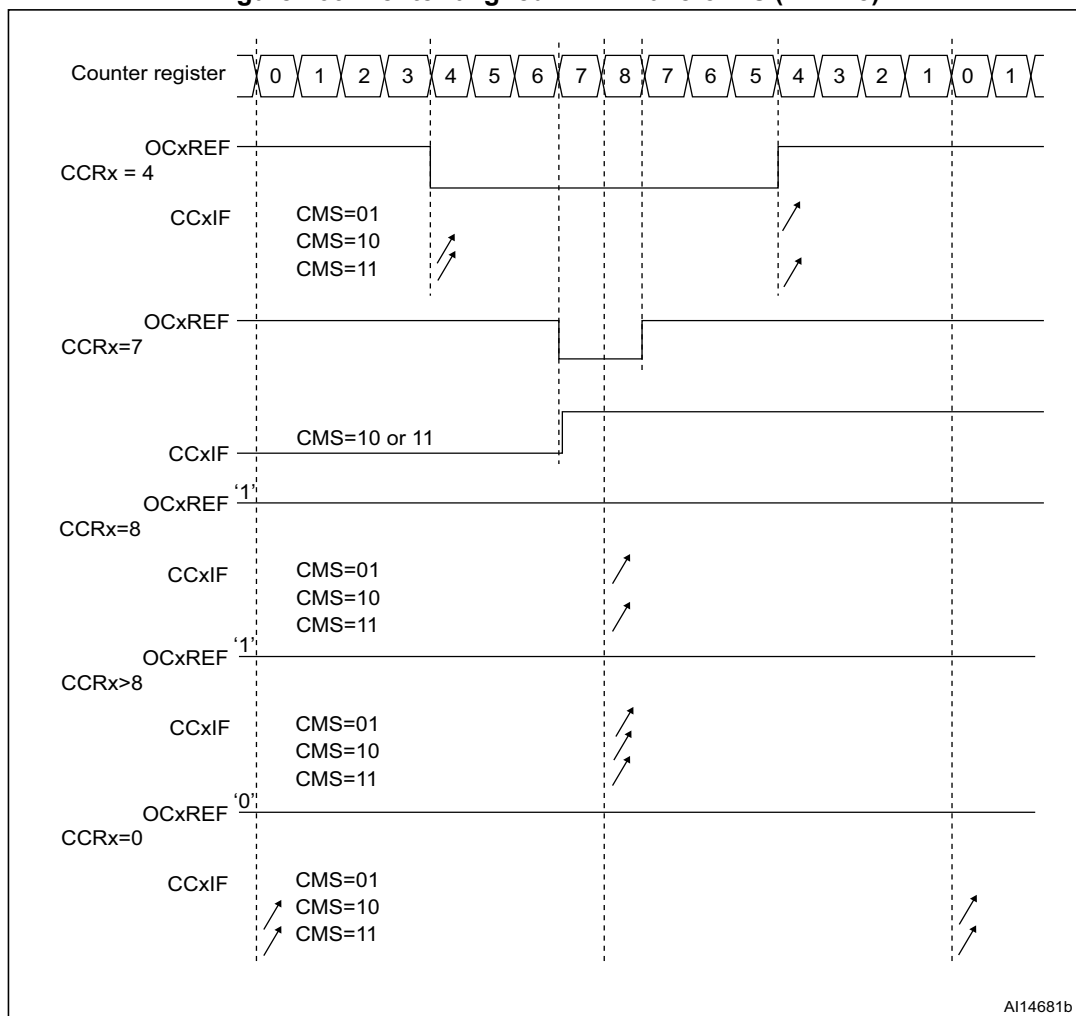
PWM center-aligned mode

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 464](#).

[Figure 166](#) 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 166. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
 - The direction is not updated if you write a value in the counter that is greater than the auto-reload value (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
 - The direction is updated if you write 0 or write the TIMx_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

20.3.11 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

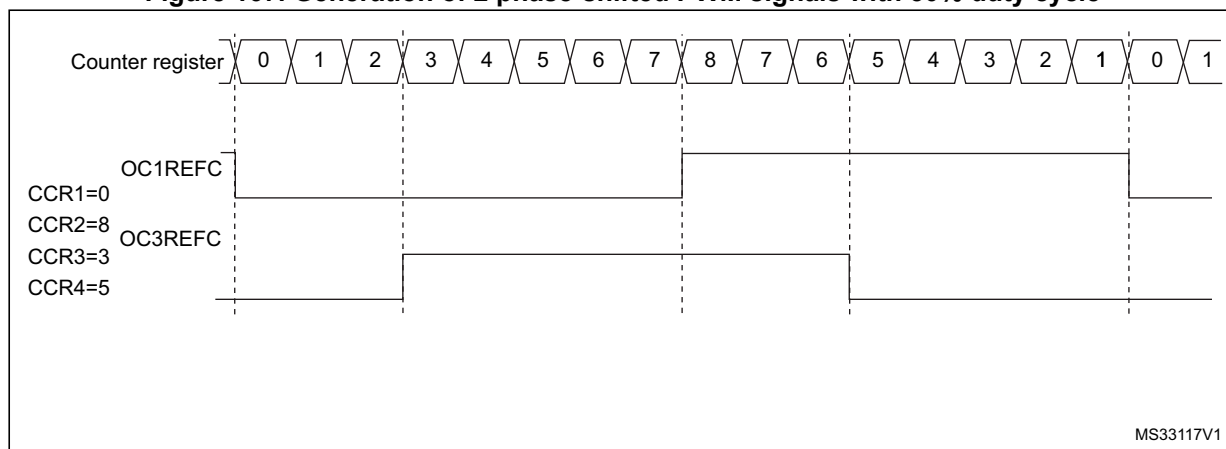
Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing '1110' (Asymmetric PWM mode 1) or '1111' (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

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 167 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 167. Generation of 2 phase-shifted PWM signals with 50% duty cycle



20.3.12 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.

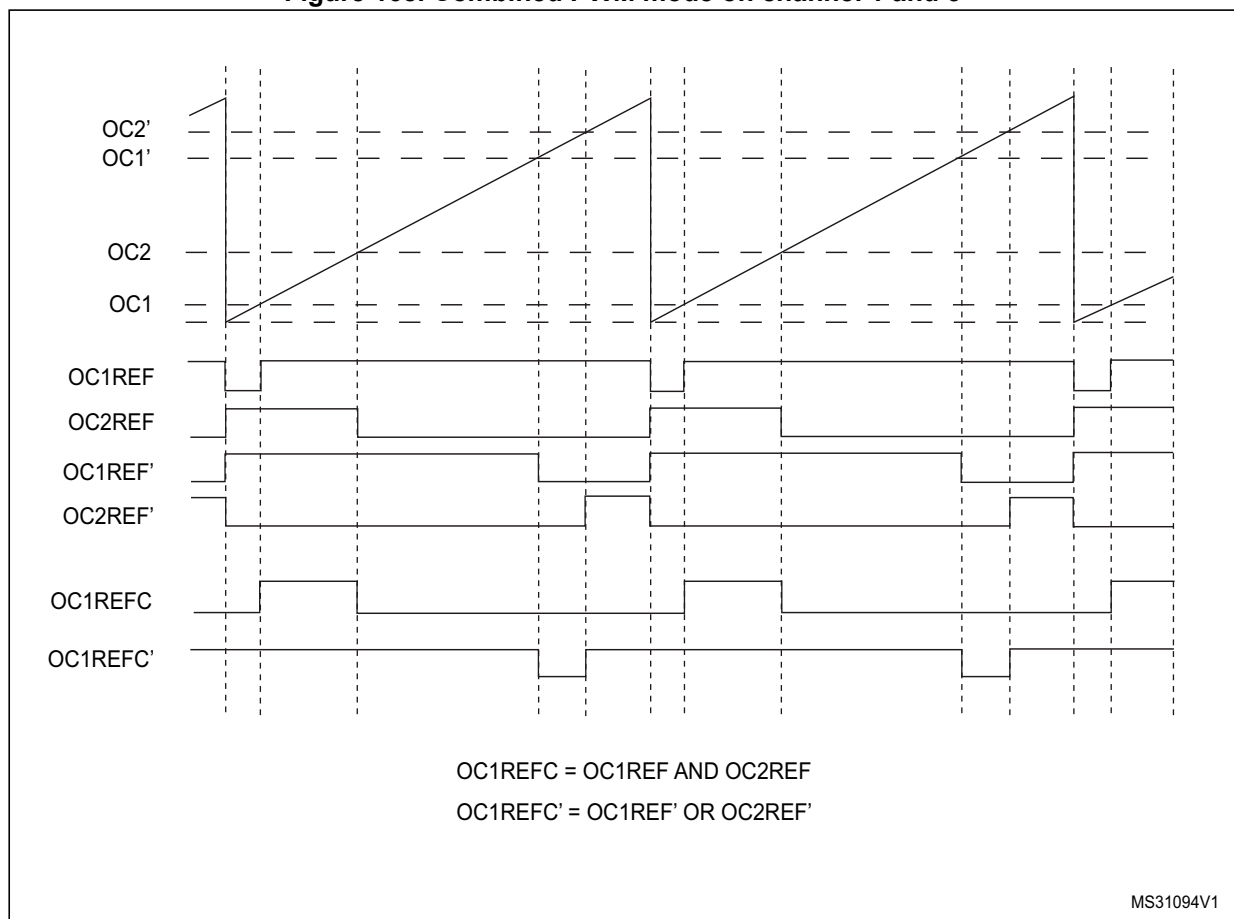
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

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 168 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 168. Combined PWM mode on channel 1 and 3



20.3.13 Combined 3-phase PWM mode

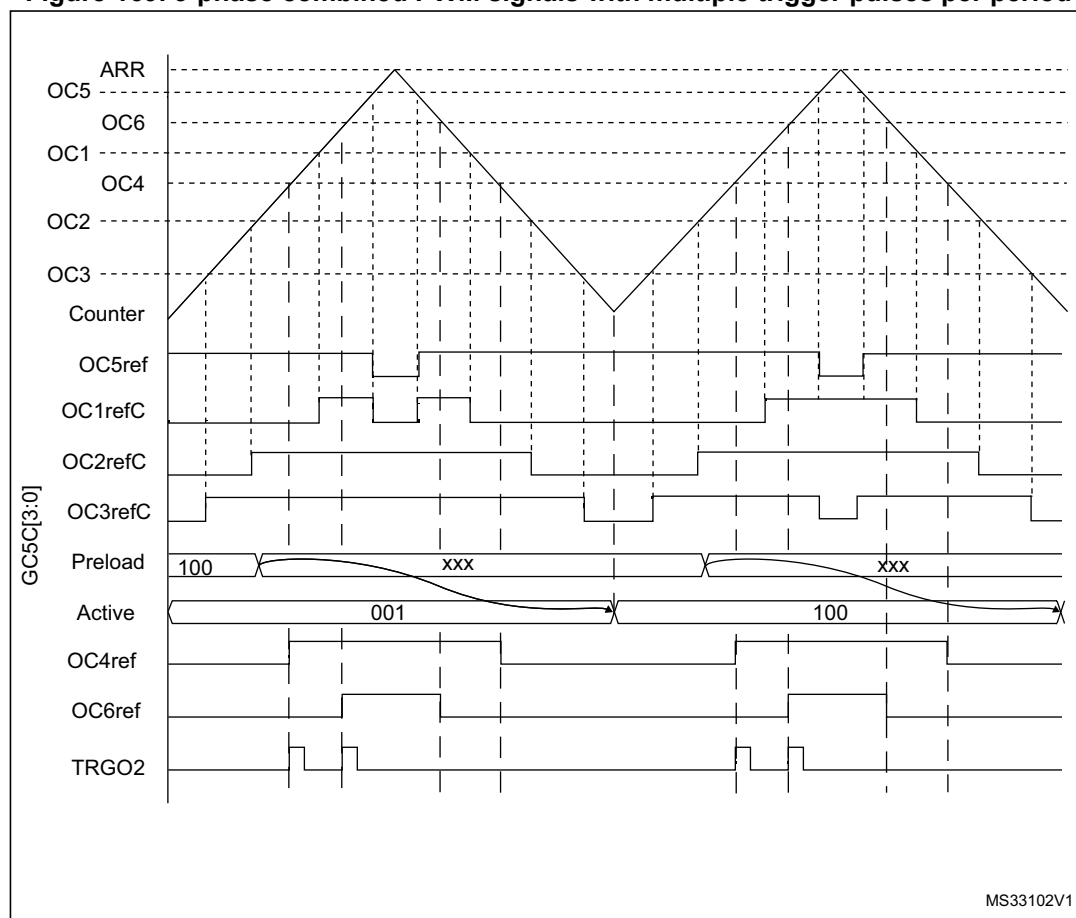
Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The

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 169. 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. Please refer to [Section 20.3.25: ADC synchronization](#) for more details.

20.3.14 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and you have to adjust it depending on the devices you have connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

You can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to [Table 116: Output control bits for complementary OCx and OCxN channels with break feature on page 531](#) 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 170. Complementary output with dead-time insertion

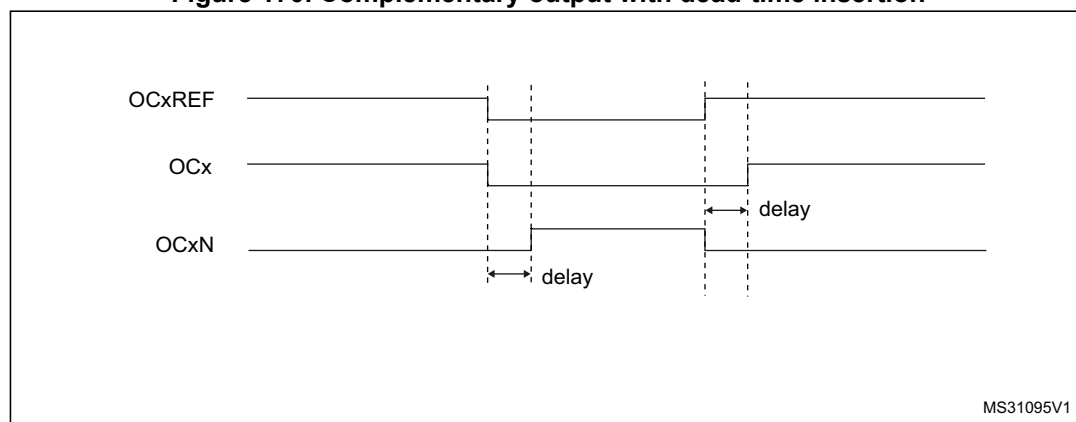
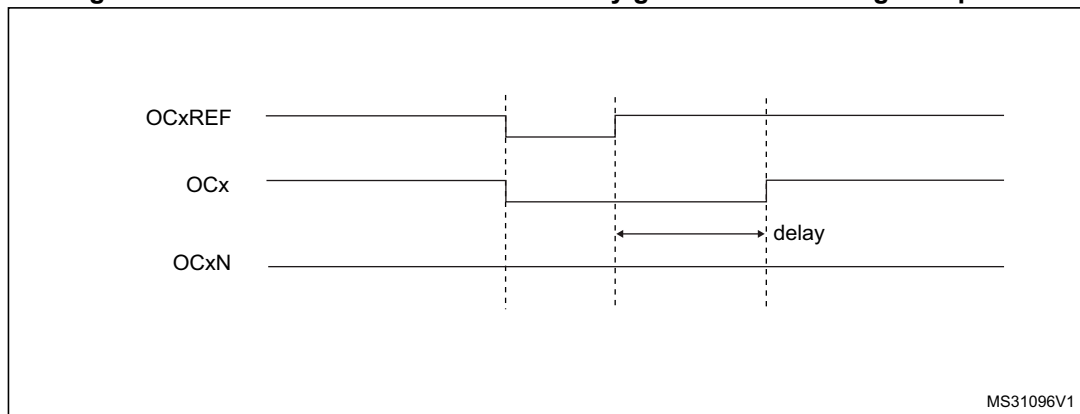
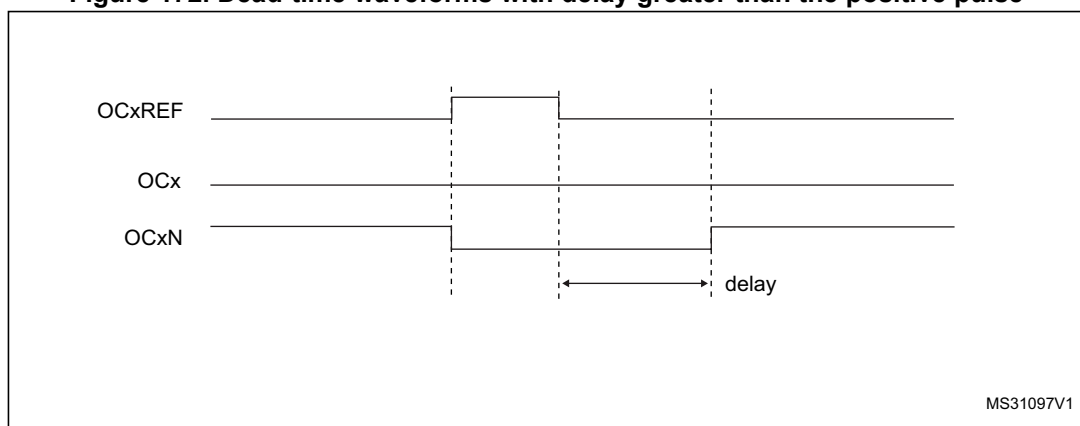


Figure 171. Dead-time waveforms with delay greater than the negative pulse**Figure 172. Dead-time waveforms with delay greater than the positive pulse**

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 20.4.18: TIM1 break and dead-time register \(TIMx_BDTR\) on page 535](#) for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows you to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

20.3.15 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.

When using the break functions, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, OISx and OISxN bits in the TIMx_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to [Table 116: Output control bits for complementary OCx and OCxN channels with break feature on page 531](#) for more details.

The source for BRK can be:

- An external source connected to the BKIN pin
- An internal source: COMP4 output

The source for BRK_ACTH can be internal only:

- A clock failure event generated by the CSS. For further information on the CSS, refer to [Section 10.2.7: Clock security system \(CSS\)](#)
- A PVD output
- SRAM parity error signal
- Cortex M4 LOCKUP (Hardfault) output
- COMPx output, x = 1, 2, and 6

The source for BRK2 can be:

- An external source connected to the BKIN2 pin
- An internal source coming from COMPx output, x = 1, 2, 4 or 6

If there are several break sources, the resulting break signal will be an OR between all the input signals.

When exiting from reset, the break circuit is disabled and the MOE bit is low. You can enable the break functions by setting the BKE and BKE2 bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BKP2 bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if you write MOE to 1 whereas it was low, you must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read the synchronous signal.

The break can be generated by any of the break inputs (BRK, BRK2, BRK_ACTH), BRK and BRK2 have:

- Programmable polarity (BKPx bit in the TIMx_BDTR register)
- Programmable enable bit (BKEx in the TIMx_BDTR register)
- Programmable filter (BKxF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

The digital filter feature is available on BRK and BRK2. It is not available on BRK_ACTH.

That means that the digital filter is:

- Available when the break source is external and comes from the external inputs BKIN/BKIN2.
- Available when the break source is internal and connected to BRK (COMP4 output) or BRK2 (all comparators' outputs)
- Not available when the break source is internal and connected to BRK_ACTH. (i.e. PVD output, SRAM parity error signal, Cortex®-M4 LOCKUP (Hardfault) output or COMPx output, x = 1, 2, 6).

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register.

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):

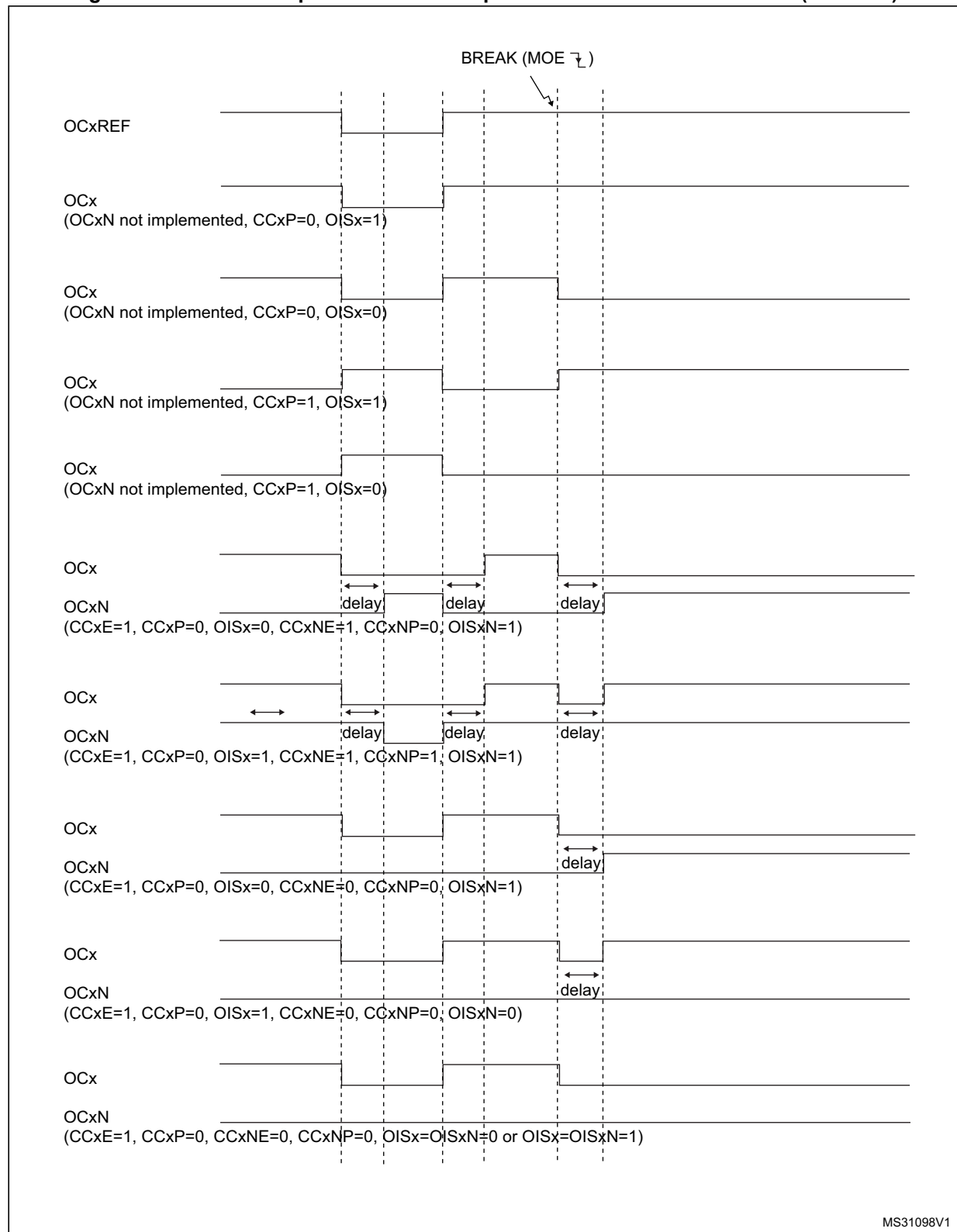
- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSS1 bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx 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 (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. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). 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 you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

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 to freeze the configuration of several parameters (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 20.4.18: TIM1 break and dead-time register \(TIMx_BDTR\) on page 535](#). The LOCK bits can be written only once after an MCU reset.

[Figure 173](#) shows an example of behavior of the outputs in response to a break.

Figure 173. Various output behavior in response to a break event on BKIN (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 113](#).

Note: **BRK2 must only be used with $OSSR = OSSl = 1$.**

Table 113. 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"> – Inactive then forced output state (after a deadtime) – Outputs disabled if $OSSl = 0$ (control taken over by GPIO logic) 	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

[Figure 174](#) gives an example of OCx and OCxN output behavior in case of active signals on BKIN and BKIN2 inputs. In this case, both outputs have active high polarities ($CCxP = CCxNP = 0$ in TIMx_CCER register).

Figure 174. PWM output state following BKIN and BKIN2 pins assertion ($OSSl=1$)

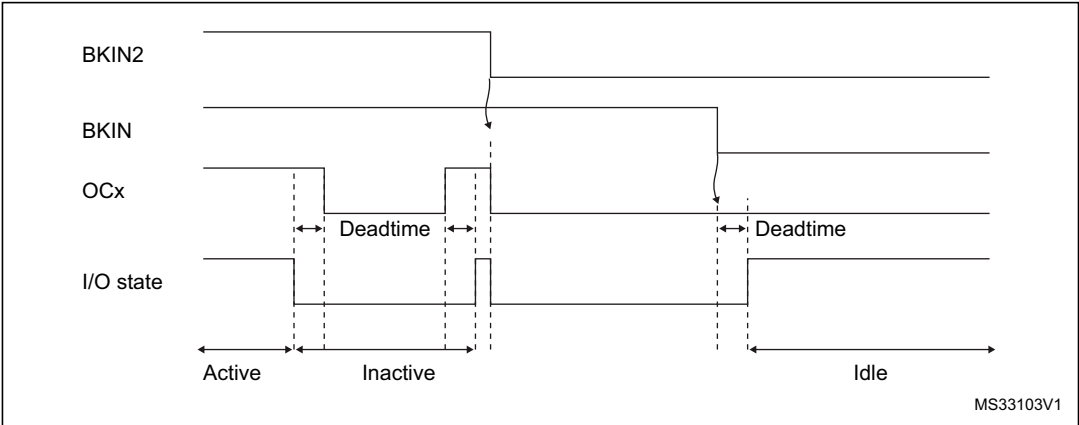
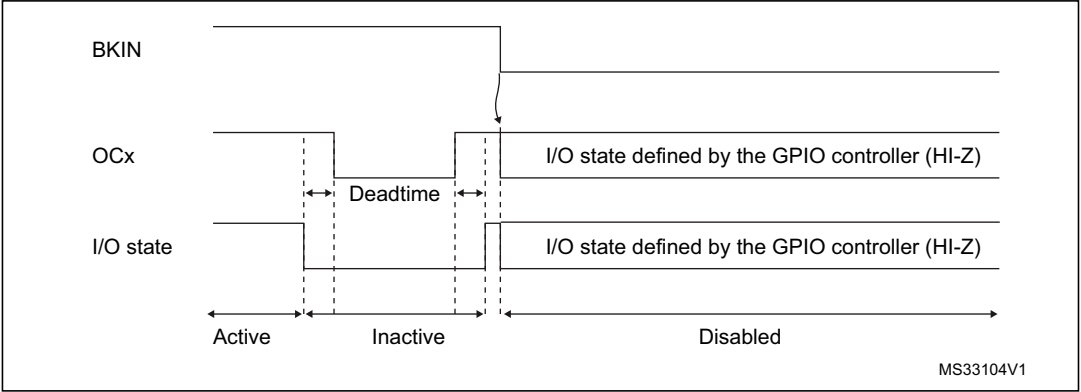


Figure 175. PWM output state following BKIN assertion (OSSI=0)



20.3.16 Clearing the OCxREF signal on an external event

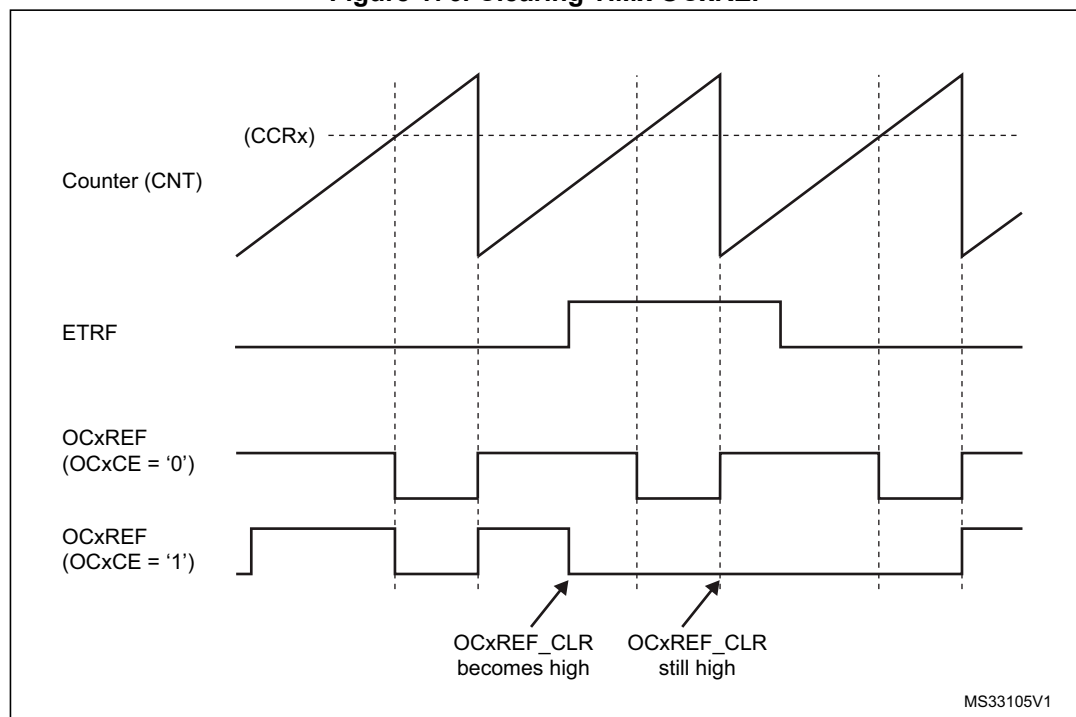
The OCxREF signal of a given channel can be cleared when a high level is applied on the OCREF_CLR_INPUT (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It 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.

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 176 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 176. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if $CCR_x > ARR$), then OCxREF is enabled again at the next counter overflow.

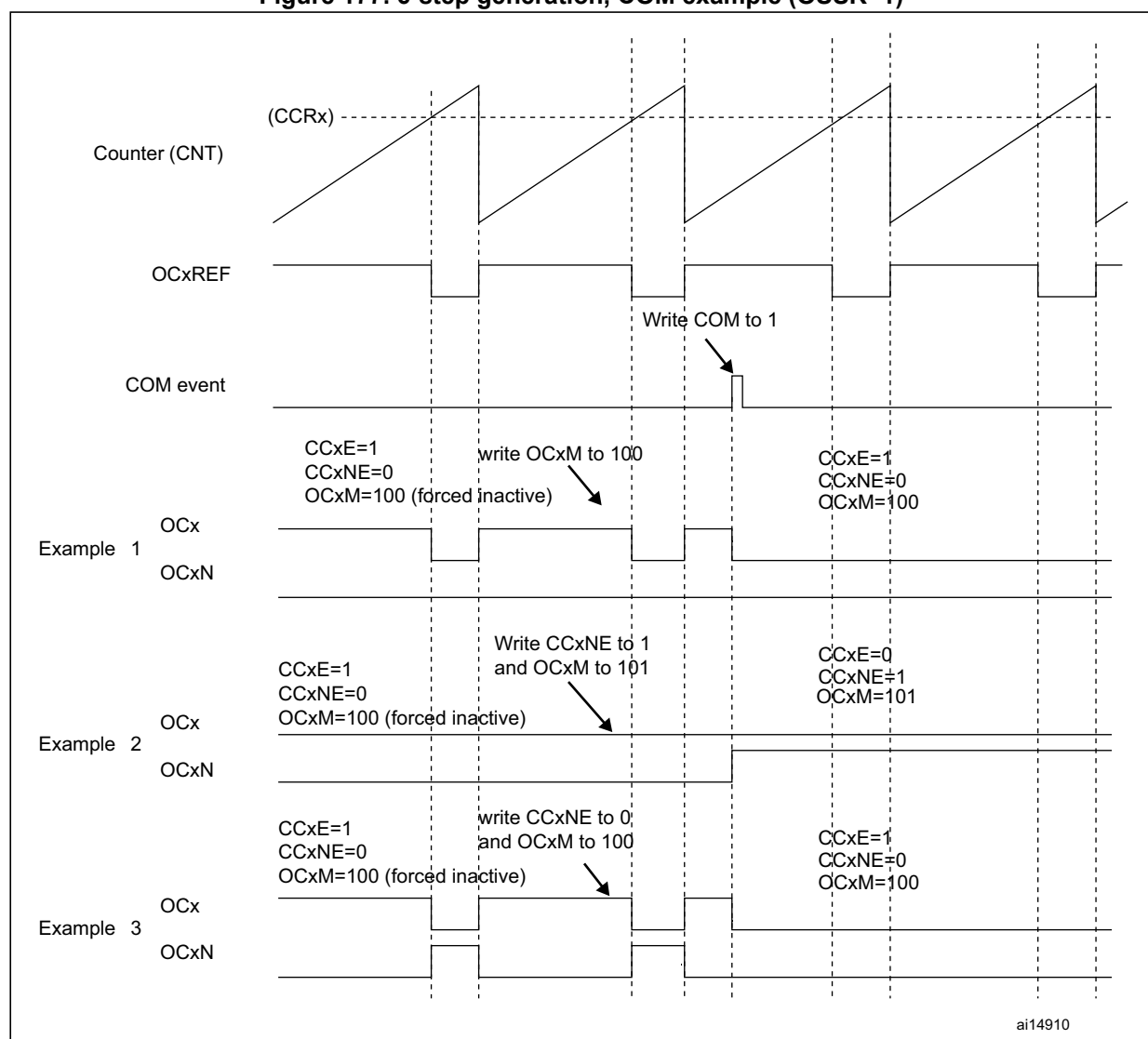
20.3.17 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus you can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The [Figure 177](#) describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 177. 6-step generation, COM example (OSSR=1)



20.3.18 One-pulse mode

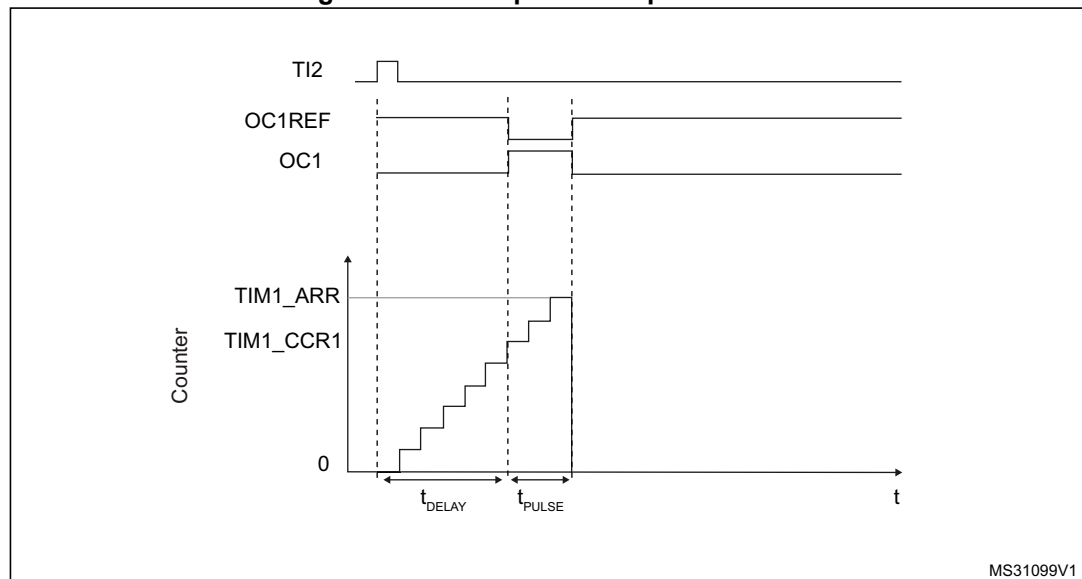
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: $CNT < CCRx \leq ARR$ (in particular, $0 < CCRx$)
- In downcounting: $CNT > CCRx$

Figure 178. Example of one pulse mode.



MS31099V1

For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

You only want 1 pulse (Single mode), so you write '1' in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay $t_{\text{DELAY min}}$ we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

20.3.19 Retriggerable one pulse mode (OPM)

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 20.3.18](#):

- 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

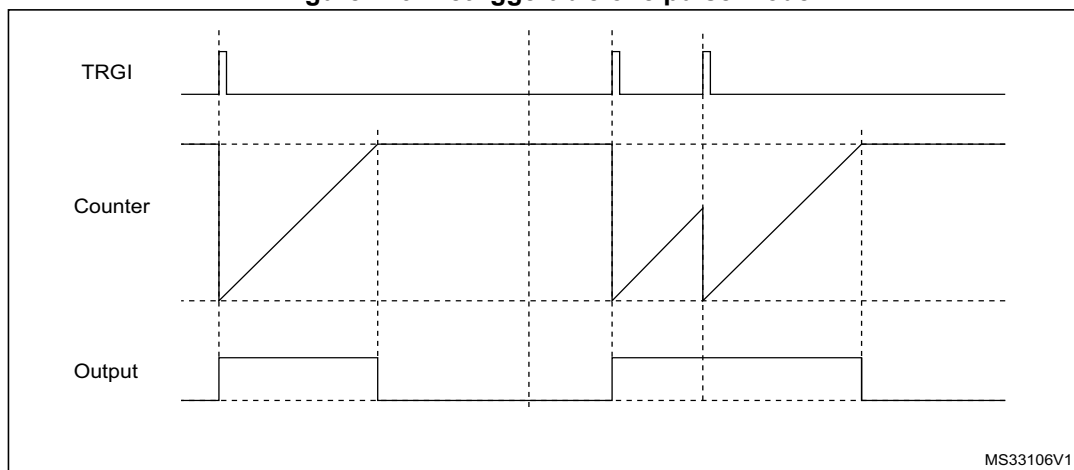
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

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, the ARR must be set to 0 (the CCRx register sets the pulse length).

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.*

In Retriggerable one pulse mode, the CCxIF flags are not significant.

Figure 179. Retriggerable one pulse mode



20.3.20 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, you can program the input filter as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an quadrature encoder. Refer to [Table 114](#). The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So you must configure TIMx_ARR before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the 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 don't switch at the same time.

Table 114. 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 180](#) gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P='0' and CC1NP='0' (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0' and CC2NP='0' (TIMx_CCER register, TI1FP2 non-inverted, TI1FP2= TI2).
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx_CR1 register, Counter enabled).

Figure 180. Example of counter operation in encoder interface mode.

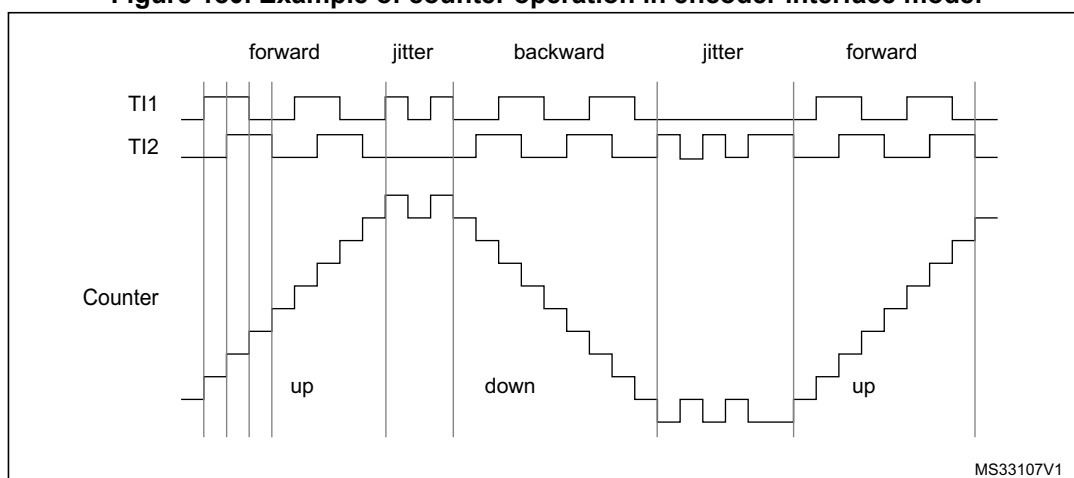
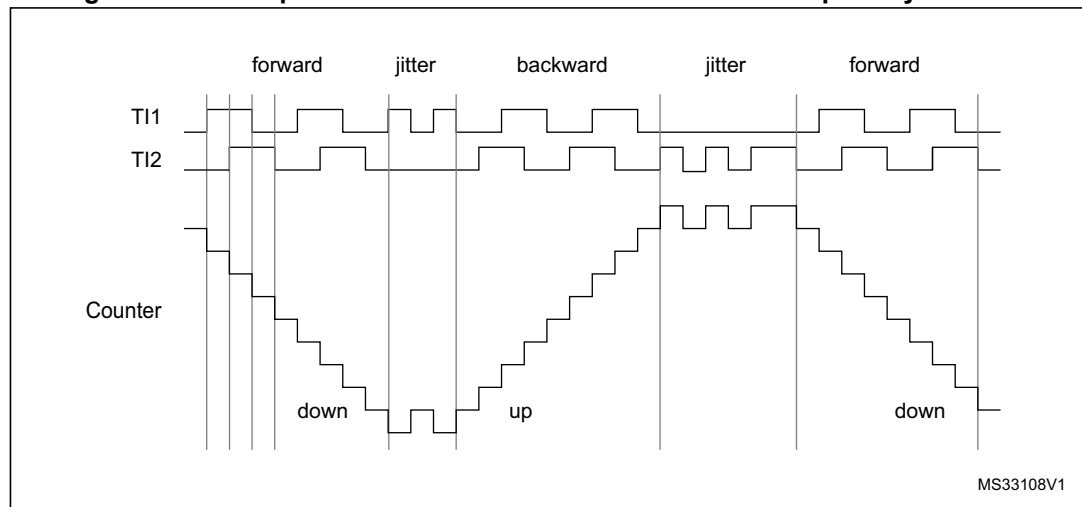


Figure 181 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 181. Example of encoder interface mode with TI1FP1 polarity inverted.



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. You can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. You can do this by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

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).

20.3.21 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).

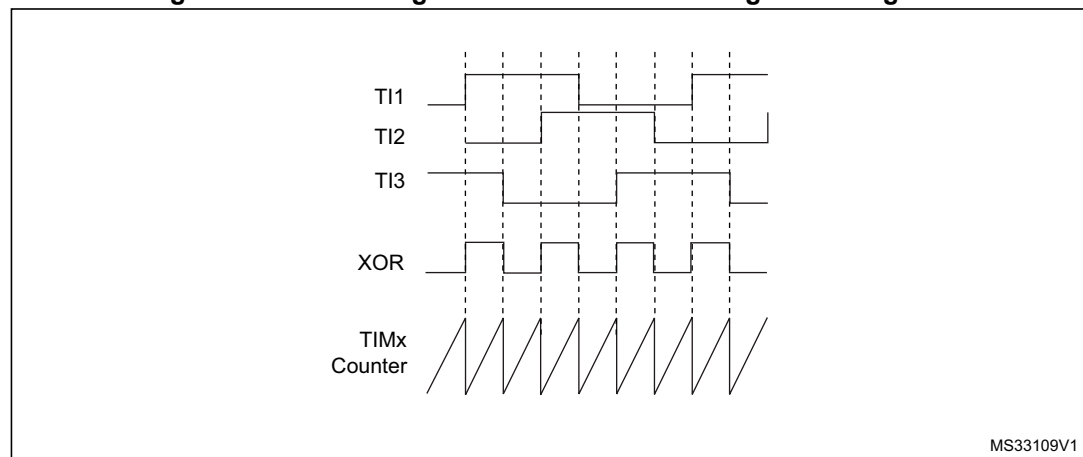
There is no latency between the UIF and UIFCPY flags assertion.

20.3.22 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 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 convenient to measure the interval between edges on two input signals, as per [Figure 182](#) below.

Figure 182. Measuring time interval between edges on 3 signals



20.3.23 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) referred to as “interfacing timer” in [Figure 183](#). 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 158: Capture/compare channel \(example: channel 1 input stage\) on page 474](#)). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1) through the TRGO output.

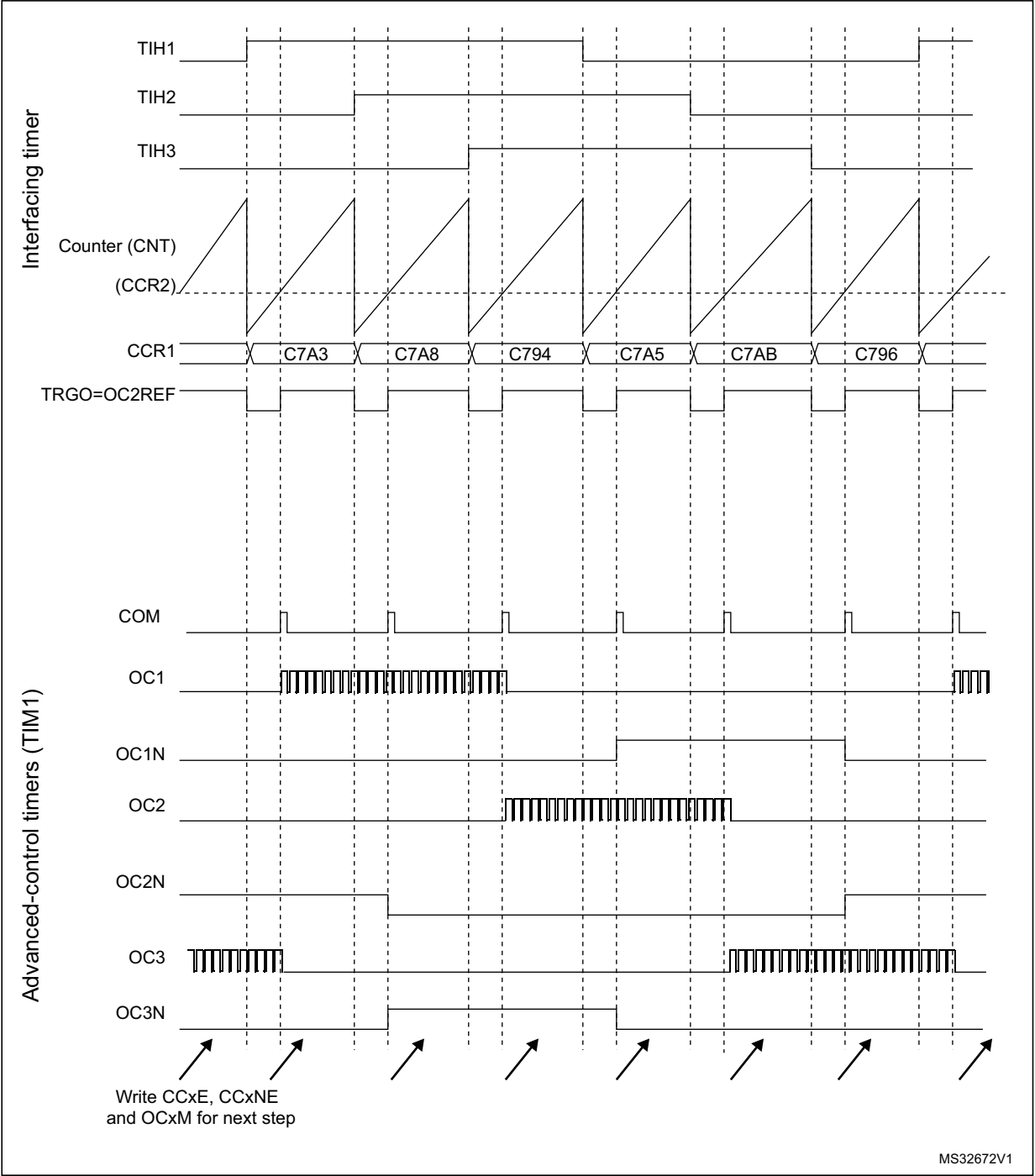
Example: you want to change the PWM configuration of your advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to '1',
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to '01'. You can also program the digital filter if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to '111' and the CC2S bits to '00' in the TIMx_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to '101',

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The [Figure 183](#) describes this example.

Figure 183. Example of Hall sensor interface



20.3.24 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. 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.

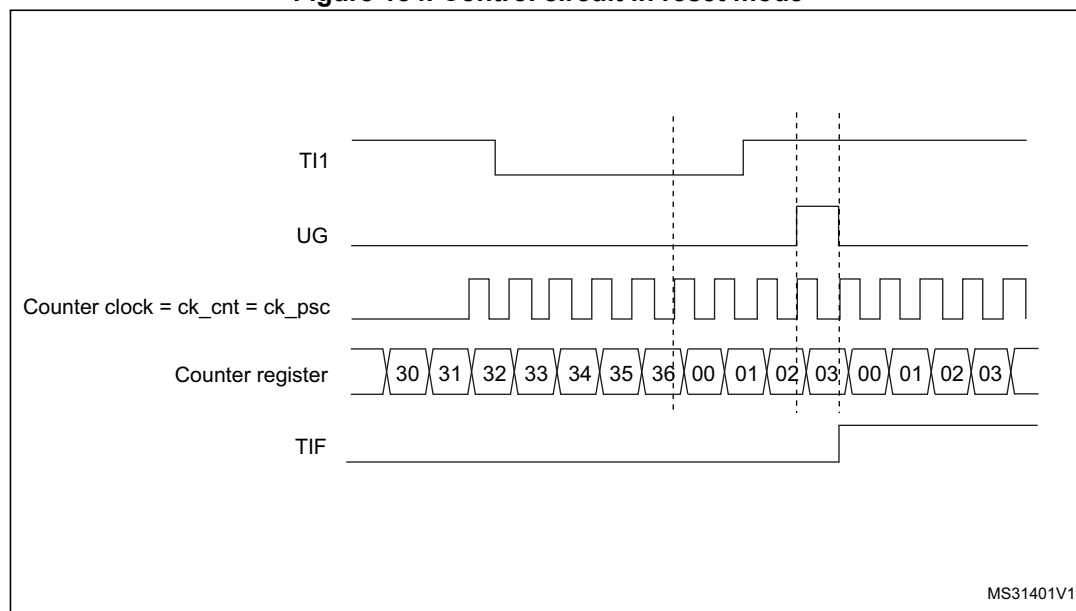
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 184. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

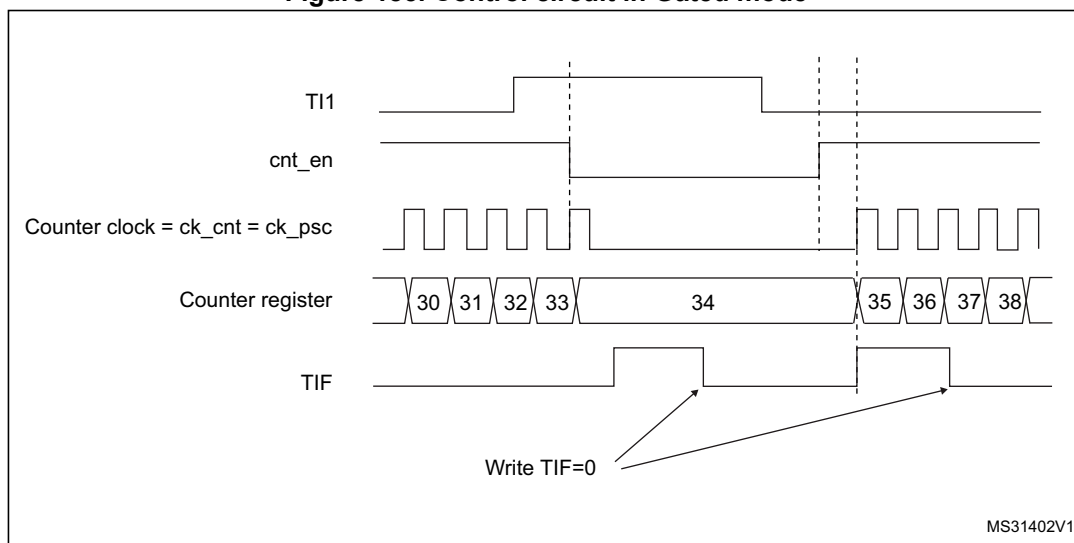
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 185. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register.

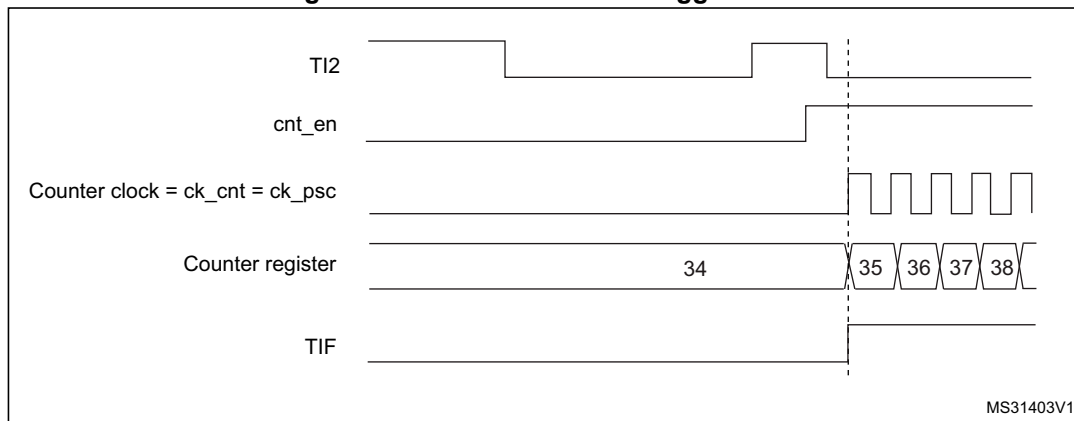
Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 186. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

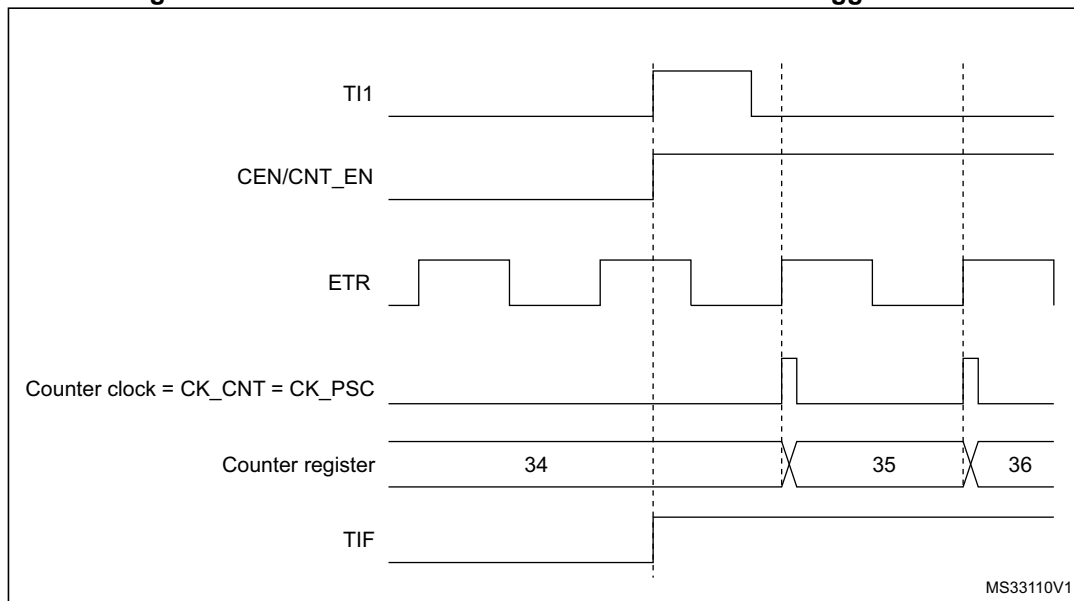
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
 - ETF = 0000: no filter
 - ETPS=00: prescaler disabled
 - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
 - IC1F=0000: no filter.
 - The capture prescaler is not used for triggering and does not need to be configured.
 - CC1S=01 in TIMx_CCMR1 register to select only the input capture source
 - CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 187. Control circuit in external clock mode 2 + trigger mode



20.3.25 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 169 on page 486](#).

20.3.26 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.

20.3.27 Debug mode

When the microcontroller enters debug mode (Cortex-M4[®]F 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 33.14.2: Debug support for timers, watchdog, bxCAN and I2C](#).

20.4 TIM1 registers

Refer to for a list of abbreviations used in register descriptions.

20.4.1 TIM1 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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, always read as 0

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 Reserved, always read as 0

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (ETR, Tlx),

00: $t_{DTS}=t_{CK_INT}$

01: $t_{DTS}=2*t_{CK_INT}$

10: $t_{DTS}=4*t_{CK_INT}$

11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered

1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled.

These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

20.4.2 TIM1 control register 2 (TIMx_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	MMS2[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:24 Reserved, always read as 0

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 OR between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on 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** - OC1REF signal is used as trigger output (TRGO2)

0101: **Compare** - OC2REF signal is used as trigger output (TRGO2)

0110: **Compare** - OC3REF signal is used as trigger output (TRGO2)

0111: **Compare** - OC4REF signal is used as trigger output (TRGO2)

1000: **Compare** - OC5REF signal is used as trigger output (TRGO2)

1001: **Compare** - OC6REF signal is used as trigger output (TRGO2)

1010: **Compare Pulse** - OC4REF rising or falling edges generate pulses on TRGO2

1011: **Compare Pulse** - OC6REF rising or falling edges generate pulses on TRGO2

1100: **Compare Pulse** - OC4REF or OC6REF rising edges generate pulses on TRGO2

1101: **Compare Pulse** - OC4REF rising or OC6REF falling edges generate pulses on TRGO2

1110: **Compare Pulse** - OC5REF or OC6REF rising edges generate pulses on TRGO2

1111: **Compare Pulse** - OC5REF rising or OC6REF falling edges generate pulses on TRGO2

Bit 19 Reserved, always read as 0

Bit 18 **OIS6**: Output Idle state 6 (OC6 output)

Refer to OIS1 bit

Bit 17 Reserved, always read as 0

Bit 16 **OIS5**: Output Idle state 5 (OC5 output)

Refer to OIS1 bit

Bit 15 Reserved, always read as 0

Bit 14 **OIS4**: Output Idle state 4 (OC4 output)

Refer to OIS1 bit

Bit 13 **OIS3N**: Output Idle state 3 (OC3N output)

Refer to OIS1N bit

Bit 12 **OIS3**: Output Idle state 3 (OC3 output)

Refer to OIS1 bit

Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)

Refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)

Refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input

1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[1:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).

100: **Compare** - OC1REF signal is used as trigger output (TRGO)

101: **Compare** - OC2REF signal is used as trigger output (TRGO)

110: **Compare** - OC3REF signal is used as trigger output (TRGO)

111: **Compare** - OC4REF signal is used as trigger output (TRGO)

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, always read as 0

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when 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.

20.4.3 TIM1 slave mode control register (TIMx_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.	Res.	Res.	Res.	Res.	Res.	SMS[3]
															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

Bit 31:17 Reserved, always read as 0

Bit 16 **SMS[3]**: Slave mode selection - bit 3

Refer to SMS description - bits2:0

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{\text{ETR}}$ is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge.

1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

Note: 1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).

2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).

3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

- 00: Prescaler OFF
- 01: ETRP frequency divided by 2
- 10: ETRP frequency divided by 4
- 11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at f_{DTS}
- 0001: $f_{SAMPLING}=f_{CK_INT}$, N=2
- 0010: $f_{SAMPLING}=f_{CK_INT}$, N=4
- 0011: $f_{SAMPLING}=f_{CK_INT}$, N=8
- 0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
- 0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
- 0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
- 0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
- 1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
- 1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
- 1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
- 1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
- 1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
- 1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
- 1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
- 1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM**: Master/slave mode

- 0: No action
- 1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

- 000: Internal Trigger 0 (ITR0)
- 001: Internal Trigger 1 (ITR1)
- 010: Internal Trigger 2 (ITR2)
- 011: Internal Trigger 3 (ITR3)
- 100: TI1 Edge Detector (TI1F_ED)
- 101: Filtered Timer Input 1 (TI1FP1)
- 110: Filtered Timer Input 2 (TI2FP2)
- 111: External Trigger input (ETRF)

See [Table 115: TIMx internal trigger connection on page 517](#) for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 **OCcs**: OCREF clear selection

This bit is used to select the OCREF clear source.

- 0: OCREF_CLR_INT is connected to the OCREF_CLR input
- 1: OCREF_CLR_INT is connected to ETRF

Bits 2:0 **SMS**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

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 TI2FP2 edge depending on TI1FP1 level.

0010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 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='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Table 115. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM15	TIM2	TIM3	TIM4 or TIM17 ⁽¹⁾
Reserved	TIM1	TIM2	TIM4 ⁽²⁾	TIM3

1. TIM1_ITR3 selection is made using bit 6 of the SYSCFG_CFGR1 register.

2. In STM32F302x6/8, TIM4 is not available but the software still needs to set TIM1_ITR3_RMP bit in SYSCFG_CFGR1 register.

20.4.4 TIM1 DMA/interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, always read as 0.

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

20.4.5 TIM1 status register (TIMx_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.	C6IF	C5IF
														rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	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

Bits 31:18 Reserved, always read as 0.

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 15:13 Reserved, always read as 0.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag

Refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag

Refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag

Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

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 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

If channel CC1 is configured as output: This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by a trigger event (refer to [Section 20.4.3: TIM1 slave mode control register \(TIMx_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx_CR1 register.

20.4.6 TIM1 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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, always read as 0.

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 to update CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/Compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

20.4.7 TIM1 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

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]
							Res.								Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
IC2F[3:0]			IC2PSC[1:0]					IC1F[3:0]			IC1PSC[1:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bits 16 **OC1M[3]**: Output Compare 1 mode - bit 3

Refer to OC1M description on bits 6:4

Bit 15 **OC2CE**: Output Compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE**: Output Compare 1 clear enable

OC1CE: Output Compare 1 Clear Enable

0: OC1Ref is not affected by the ETRF Input

1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

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.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample T11 input and the length of the digital filter applied to T11. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
 0001: $f_{SAMPLING}=f_{CK_INT}$, $N=2$
 0010: $f_{SAMPLING}=f_{CK_INT}$, $N=4$
 0011: $f_{SAMPLING}=f_{CK_INT}$, $N=8$
 0100: $f_{SAMPLING}=f_{DTS}/2$, $N=6$
 0101: $f_{SAMPLING}=f_{DTS}/2$, $N=8$
 0110: $f_{SAMPLING}=f_{DTS}/4$, $N=6$
 0111: $f_{SAMPLING}=f_{DTS}/4$, $N=8$
 1000: $f_{SAMPLING}=f_{DTS}/8$, $N=6$
 1001: $f_{SAMPLING}=f_{DTS}/8$, $N=8$
 1010: $f_{SAMPLING}=f_{DTS}/16$, $N=5$
 1011: $f_{SAMPLING}=f_{DTS}/16$, $N=6$
 1100: $f_{SAMPLING}=f_{DTS}/16$, $N=8$
 1101: $f_{SAMPLING}=f_{DTS}/32$, $N=5$
 1110: $f_{SAMPLING}=f_{DTS}/32$, $N=6$
 1111: $f_{SAMPLING}=f_{DTS}/32$, $N=8$

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as $CC1E=0$ (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
 01: capture is done once every 2 events
 10: capture is done once every 4 events
 11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
 01: CC1 channel is configured as input, IC1 is mapped on T11
 10: CC1 channel is configured as input, IC1 is mapped on T12
 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).

20.4.8 TIM1 capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

Refer to the above CCMR1 register description.

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]
							Res.								Res.
							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]	
IC4F[3:0]			IC4PSC[1:0]		IC3F[3:0]			IC3PSC[1:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC4M[3]**: Output Compare 4 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bit 16 **OC3M[3]**: Output Compare 3 mode - bit 3

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

20.4.9 TIM1 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, always read as 0.

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, always read as 0.

Bit 17 **CC5P**: Capture/Compare 5 output polarity

Refer to CC1P description

Bit 16 **CC5E**: Capture/Compare 5 output enable

Refer to CC1E description

- Bits 15 **CC4NP**: Capture/Compare 4 complementary output polarity
Refer to CC1NP description
- Bit 14 Reserved, always read as 0.
- 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).
Note: 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.
Note: 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

CC1 channel configured as output:

- 0: OC1 active high
- 1: OC1 active low

CC1 channel configured as input: CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

00: 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).

01: 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).

10: reserved, do not use this configuration.

11: 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.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Note: 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

CC1 channel configured as output:

- 0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
- 1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

- 0: Capture disabled.
- 1: Capture enabled.

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 116. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states ⁽¹⁾	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	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) OCx=CCxP, OCxN=CCxNP	
	1		0	0	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered). Then (this is valid only if BRK is triggered), if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state (may cause a short circuit when driving switches in half-bridge configuration). Note: BRK2 can only be used if OSSI = OSSR = 1.	
			0	1		
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

20.4.10 TIM1 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY	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 the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, always read as 0.

Bits 15:0 **CNT[15:0]**: Counter value

20.4.11 TIM1 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to $f_{CK_PSC} / (PSC[15:0] + 1)$.

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

20.4.12 TIM1 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 20.3.1: Time-base unit on page 455](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

20.4.13 TIM1 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000

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

20.4.14 TIM1 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:: CR1 is the counter value transferred by the last input capture 1 event (IC1).

20.4.15 TIM1 capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_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).

20.4.16 TIM1 capture/compare register 3 (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC3 output.

If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (IC3).

20.4.17 TIM1 capture/compare register 4 (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_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).

20.4.18 TIM1 break and dead-time register (TIMx_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.	Res.	Res.	Res.	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
						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
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Note: As the bits 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 31:26 Reserved.

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 = OSSI = 1.

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bit-field defines the frequency used to sample BRK2 input and the length of the digital filter applied to BRK2. The digital filter is made of an event counter in which N 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 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 20.4.9: TIM1 capture/compare enable register \(TIMx_CCER\) on page 528](#)).

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

0: Break inputs (BRK and CCS clock failure event) disabled

1: Break inputs (BRK and CCS clock failure event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details ([Section 20.4.9: TIM1 capture/compare enable register \(TIMx_CCER\) on page 528](#)).

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 20.4.9: TIM1 capture/compare enable register \(TIMx_CCER\) on page 528](#)).

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 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 $T_{dtg}=2xt_{DTS}$.

DTG[7:5]=110 => DT=(32+DTG[4:0])x t_{dtg} with $T_{dtg}=8xt_{DTS}$.

DTG[7:5]=111 => DT=(32+DTG[4:0])x t_{dtg} with $T_{dtg}=16xt_{DTS}$.

Example if $T_{DTS}=125\text{ns}$ (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

20.4.19 TIM1 DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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, always read as 0

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer

00001: 2 transfers

00010: 3 transfers

...

10001: 18 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

– If DBL = 7 bytes and DBA = TIM2_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 will be 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 you configure the DMA Data Size in half-words, 16-bit data will be transferred to each of the 7 registers.
- If you configure the DMA Data Size in bytes, the data will also be transferred to 7 registers: the first register will contain the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, you also have to specify the size of data transferred by DMA.

Bits 7:5 Reserved, always read as 0

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

...

20.4.20 TIM1 DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

20.4.21 TIM1 capture/compare mode register 3 (TIMx_CCMR3)

Address offset: 0x54

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

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
OC6CE	OC6M[2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Output compare mode

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC6M[3]**: Output Compare 6 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bit 16 **OC5M[3]**: Output Compare 5 mode - bit 3

Bit 15 **OC6CE**: Output compare 6 clear enable

Bits 14:12 **OC6M**: Output compare 6 mode

Bit 11 **OC6PE**: Output compare 6 preload enable

Bit 10 **OC6FE**: Output compare 6 fast enable

Bits 9:8 Reserved, always read as 0.

Bit 7 **OC5CE**: Output compare 5 clear enable

Bits 6:4 **OC5M**: Output compare 5 mode

Bit 3 **OC5PE**: Output compare 5 preload enable

Bit 2 **OC5FE**: Output compare 5 fast enable

Bits 1:0 Reserved, always read as 0.

20.4.22 TIM1 capture/compare register 5 (TIMx_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.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group Channel 5 and Channel 3

Distortion on Channel 3 output:

0: No effect of OC5REF on OC3REFC

1: OC3REFC is the logical AND of OC3REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMx_CCMR2).

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 TIMx_CCMR1).

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 TIMx_CCMR1).

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.

20.4.23 TIM1 capture/compare register 6 (TIMx_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.

20.4.24 TIM1 option registers (TIMx_OR)

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.	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.	TIM1_ETR_ADC1_RMP	
														rw	rw

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

Bits 1:0 **TIM1_ETR_ADC1_RMP[1:0]**: TIM1_ETR_ADC1 remapping capability

00: TIM1_ETR is not connected to any AWD

01: TIM1_ETR is connected to ADC1 AWD1

10: TIM1_ETR is connected to ADC1 AWD2

11: TIM1_ETR is connected to ADC1 AWD3

Note: ADC1 AWD is 'ORed' with the other TIM1_ETR source signals. It is consequently necessary to disable by software other sources (input pins).

Note:

20.4.25 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 117. TIM1 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	TIMx_CR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UIFREMAP	Res	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN	
	Reset value																					0		0	0	0	0	0	0	0	0	0	0	
0x04	TIMx_CR2	Res	Res	Res	Res	Res	Res	Res	Res	MMS2[3:0]					OIS6	Res	OIS5	Res	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TIS	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	0	
0x08	TIMx_SMCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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	
0x0C	TIMx_DIER	Res	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	TIMx_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	C6IF	C5IF	Res	Res	Res	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	B1F	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																0	0				0	0	0	0	0	0	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	B2G	BG	TG	COM	CC4G	CC3G	CC2G	CC1G	UG	
	Reset value																0	0							0	0	0	0	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]			OC2PE	OC2FE	CC2S [1:0]	OC1CE	OC1M [2:0]			OC1PE	OC1FE	IC1PSC [1:0]	CC1S [1:0]		
	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	IC2F[3:0]			IC2PSC [1:0]	CC2S [1:0]	IC1F[3:0]			IC1PSC [1:0]	CC1S [1:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x1C	TIMx_CCMR2 Output Compare mode	Res	Res	Res	Res	Res	Res	Res	OC4M[3]	Res	Res	Res	Res	Res	Res	Res	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]	OC3CE	OC3M [2:0]			OC3PE	OC3FE	IC3PSC [1:0]	CC3S [1:0]		
	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	IC4F[3:0]			IC4PSC [1:0]	CC4S [1:0]	IC3F[3:0]			IC3PSC [1:0]	CC3S [1:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20	TIMx_CCER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC6P	CC6E	Res	Res	Res	CC5P	CC5E	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	
	Reset value											0	0				0	0																

Table 117. TIM1 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	TIMx_CNT	UICPY	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	TIMx_PSC	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	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	0	0	0	0	0	0
0x30	TIMx_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	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
0x38	TIMx_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	0
0x3C	TIMx_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	0
0x40	TIMx_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	0
0x44	TIMx_BDTR	Res	Res	Res	Res	Res	BK2P	BK2E	BK2F[3:0]			BKF[3:0]			MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]	DT[7:0]											
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	DBL[4:0]				Res	Res	Res	DBA[4:0]					
	Reset value																				0	0	0	0				0	0	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
0x54	TIMx_CCMR3 Output Compare mode	Res	Res	Res	Res	Res	Res	OC6M[3]	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	0			
0x58	TIMx_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	0

Table 117. TIM1 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5C	TIMx_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	TIMx_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	TIM1_ETR_A DC1_RMP
	Reset value																															0	0

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

21 General-purpose timers (TIM2/TIM3/TIM4)

21.1 TIM2/TIM3/TIM4 introduction

The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 21.3.19](#).

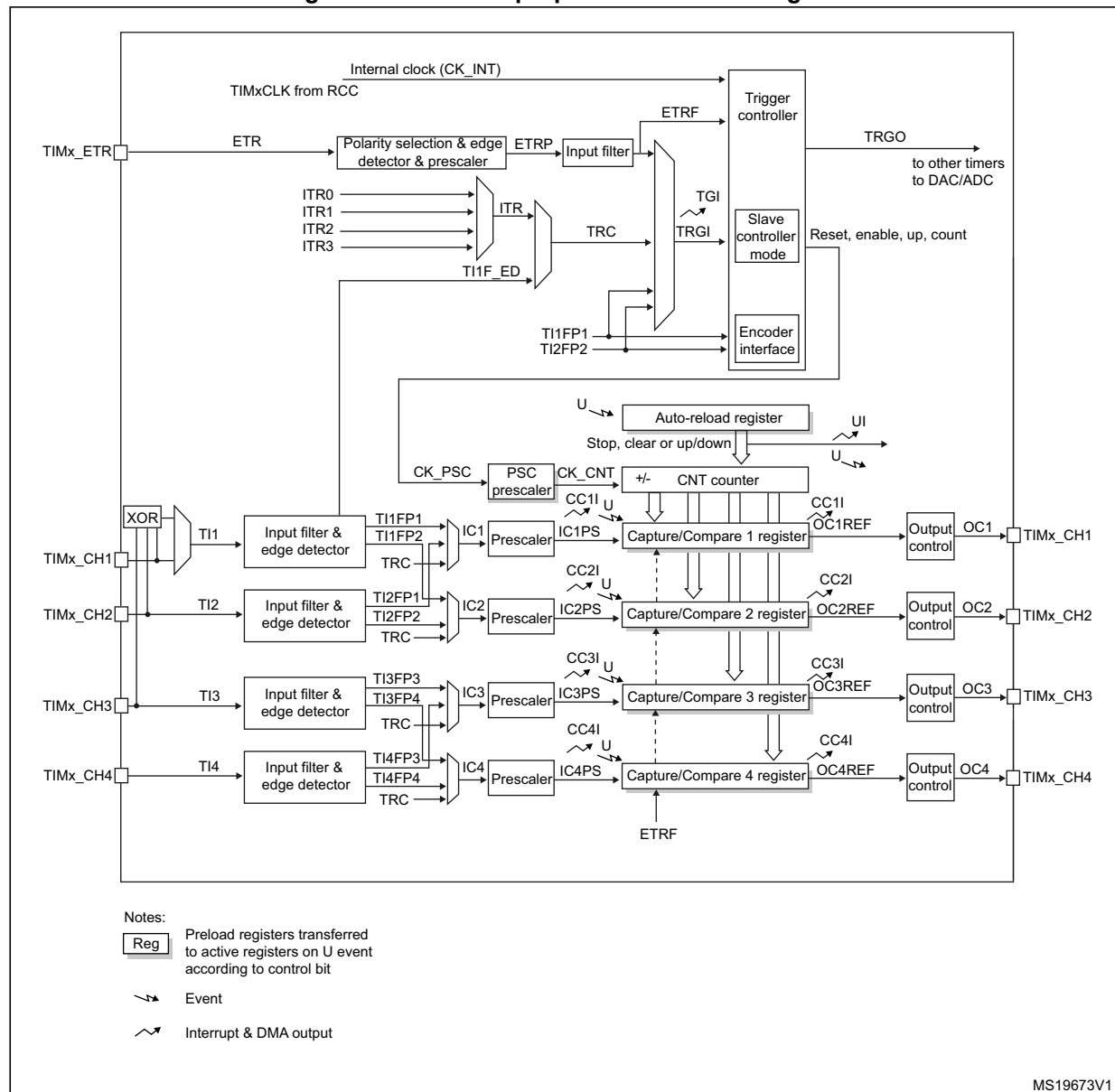
Note: TIM3 and TIM4 are available only on STM32F302xB/C/D/E devices.

21.2 TIM2/TIM3/TIM4 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3 and TIM4) 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

Figure 188. General-purpose timer block diagram



21.3 TIM2/TIM3/TIM4 functional description

21.3.1 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC):
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 189 and *Figure 190* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 189. Counter timing diagram with prescaler division change from 1 to 2

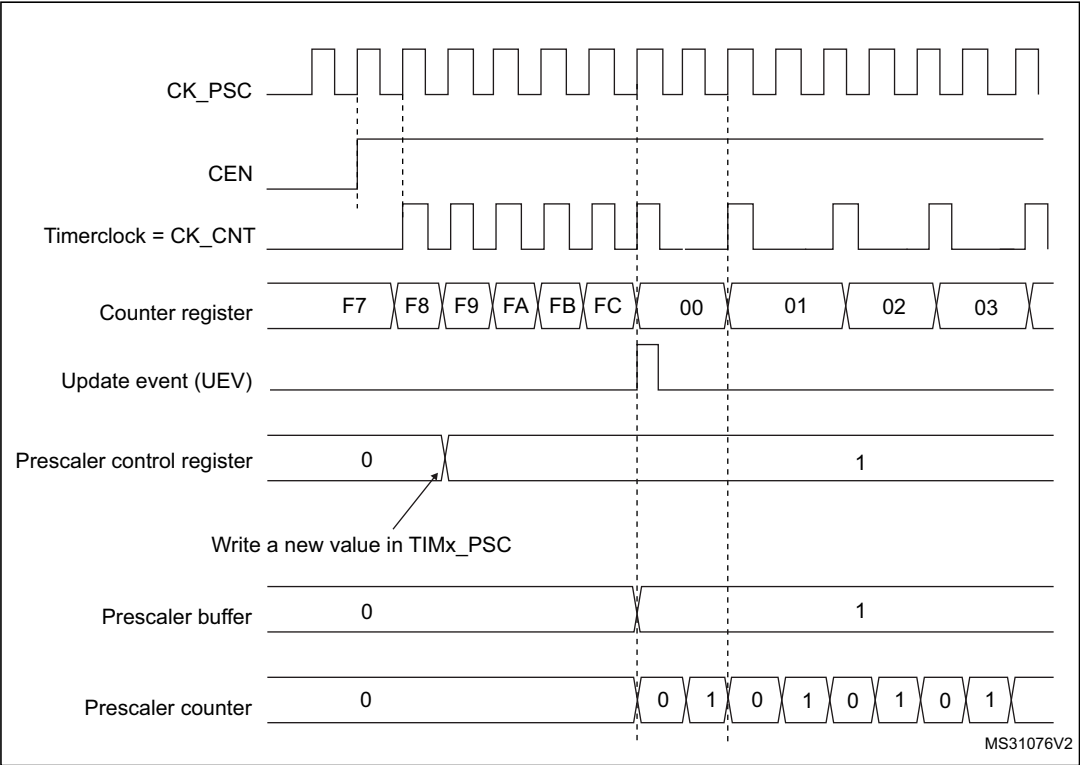
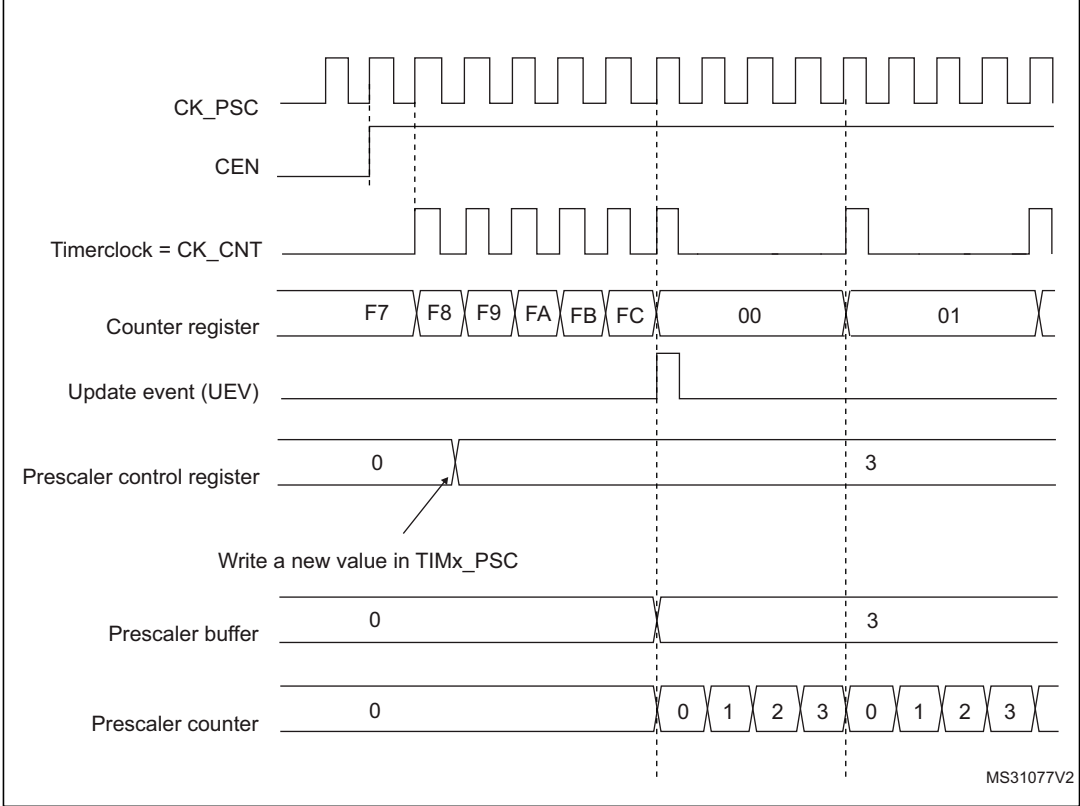


Figure 190. Counter timing diagram with prescaler division change from 1 to 4



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.

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 191. Counter timing diagram, internal clock divided by 1

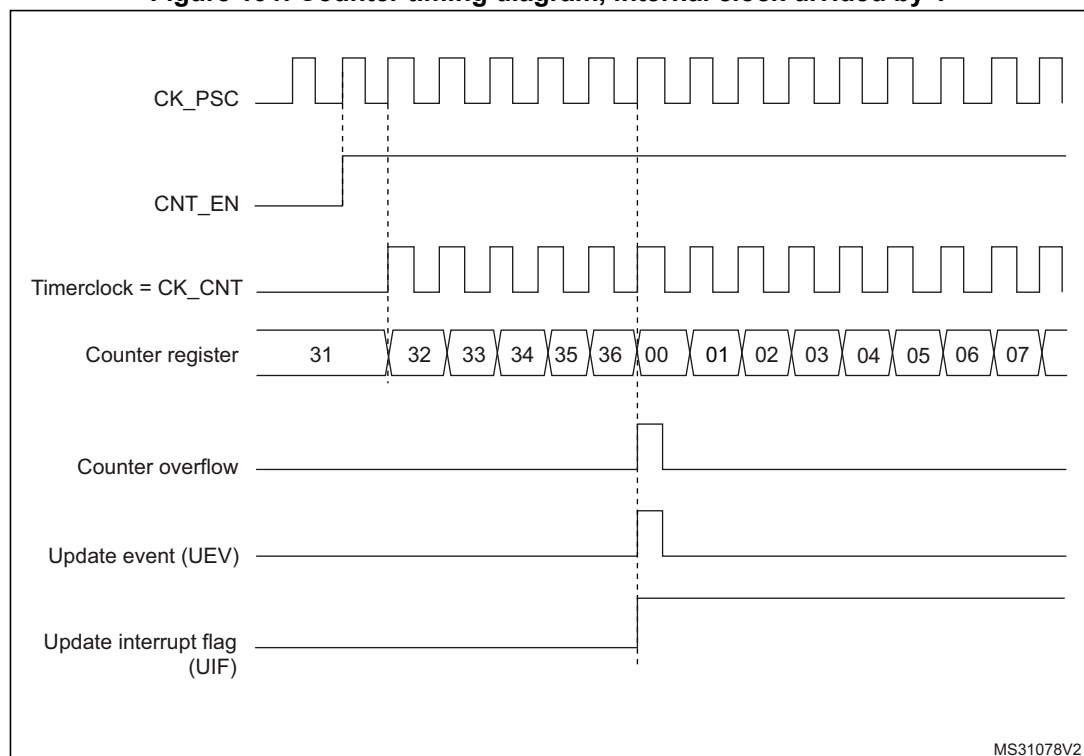


Figure 192. Counter timing diagram, internal clock divided by 2

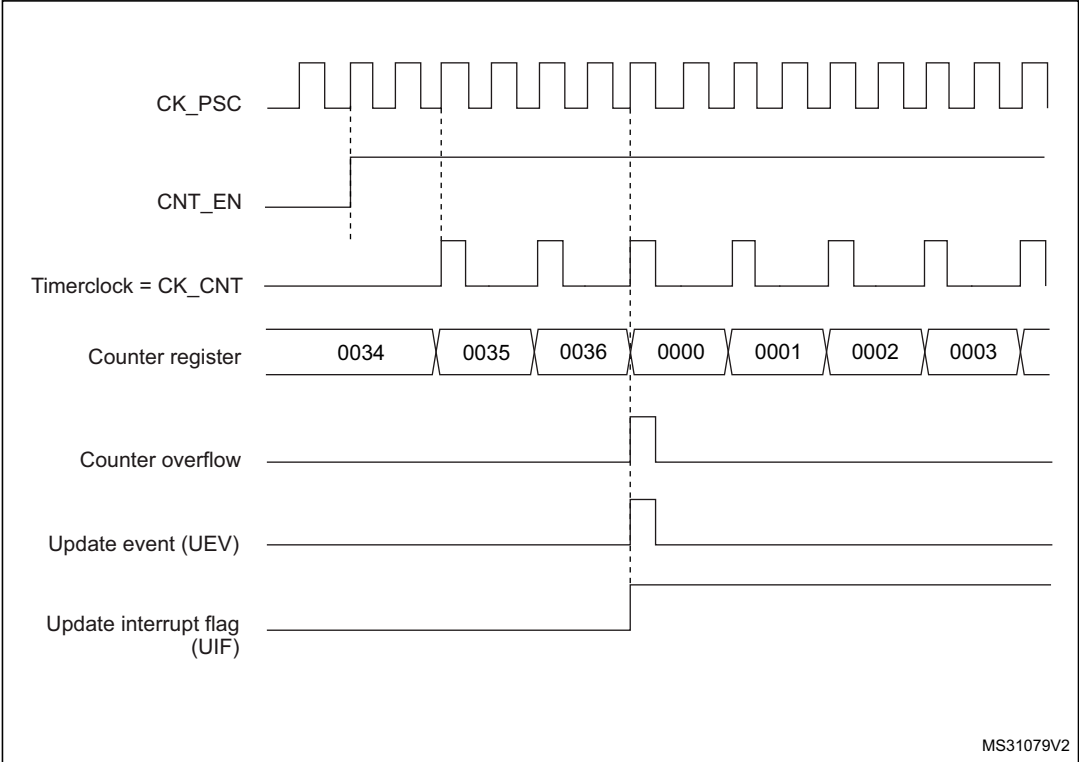


Figure 193. Counter timing diagram, internal clock divided by 4

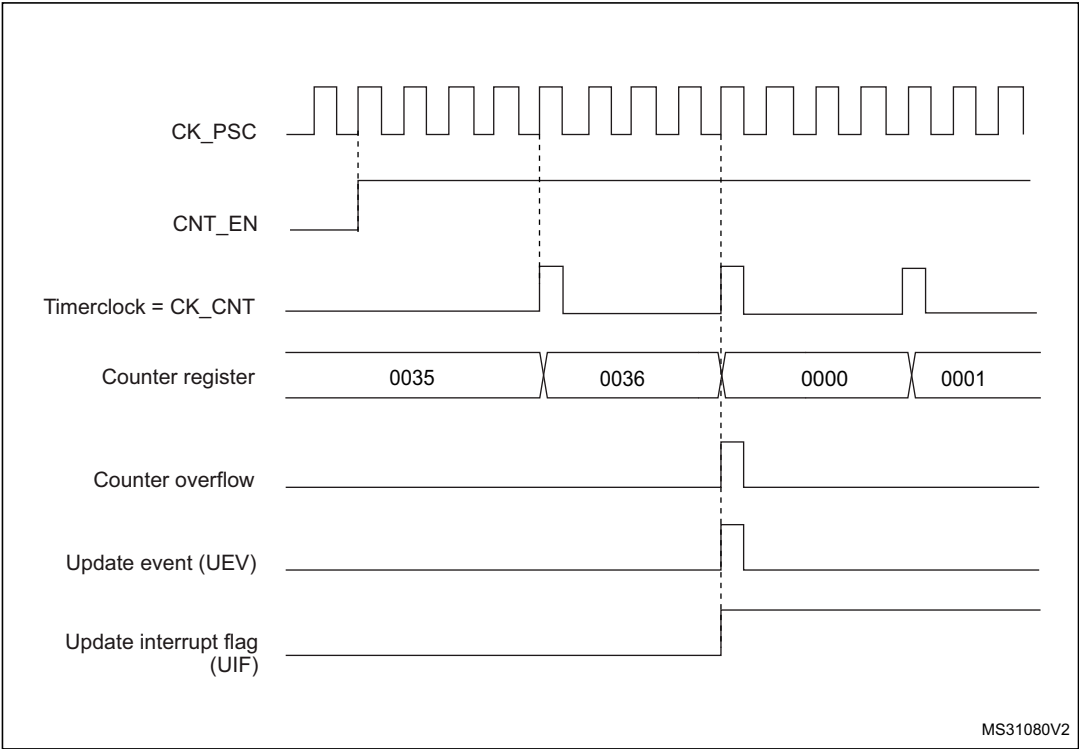


Figure 194. Counter timing diagram, internal clock divided by N

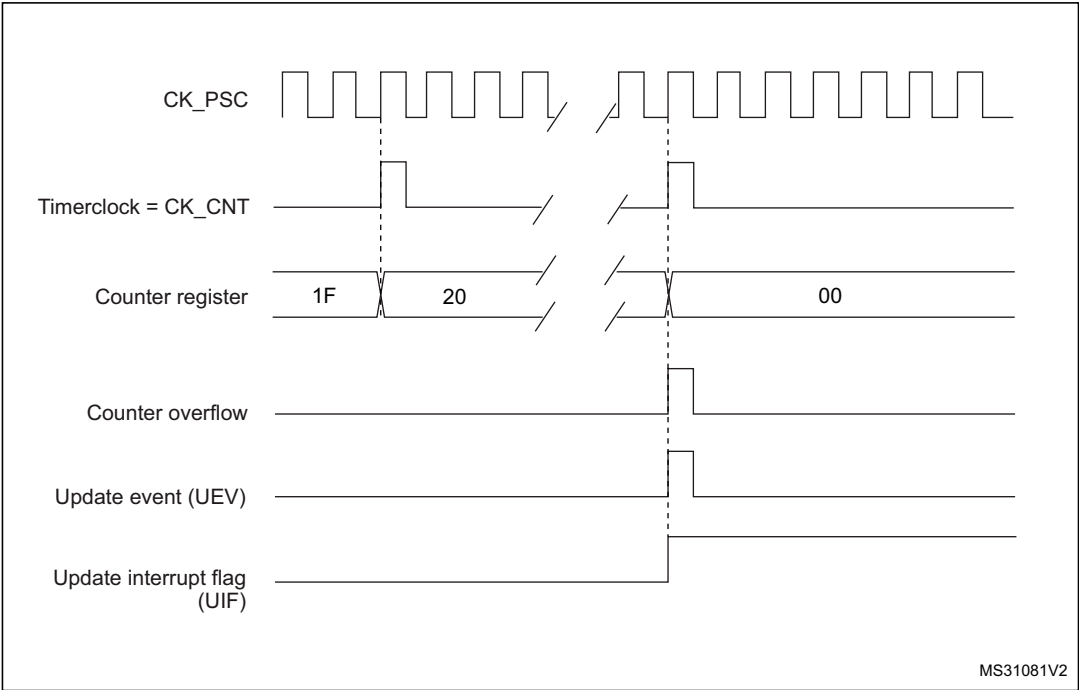


Figure 195. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

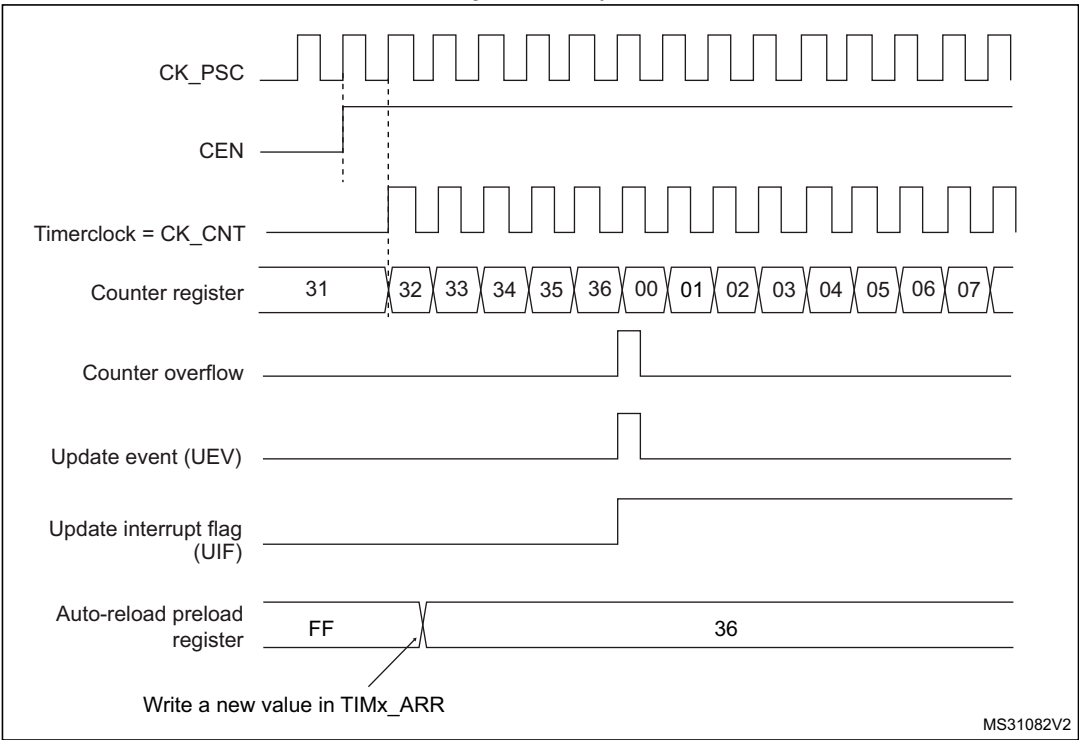
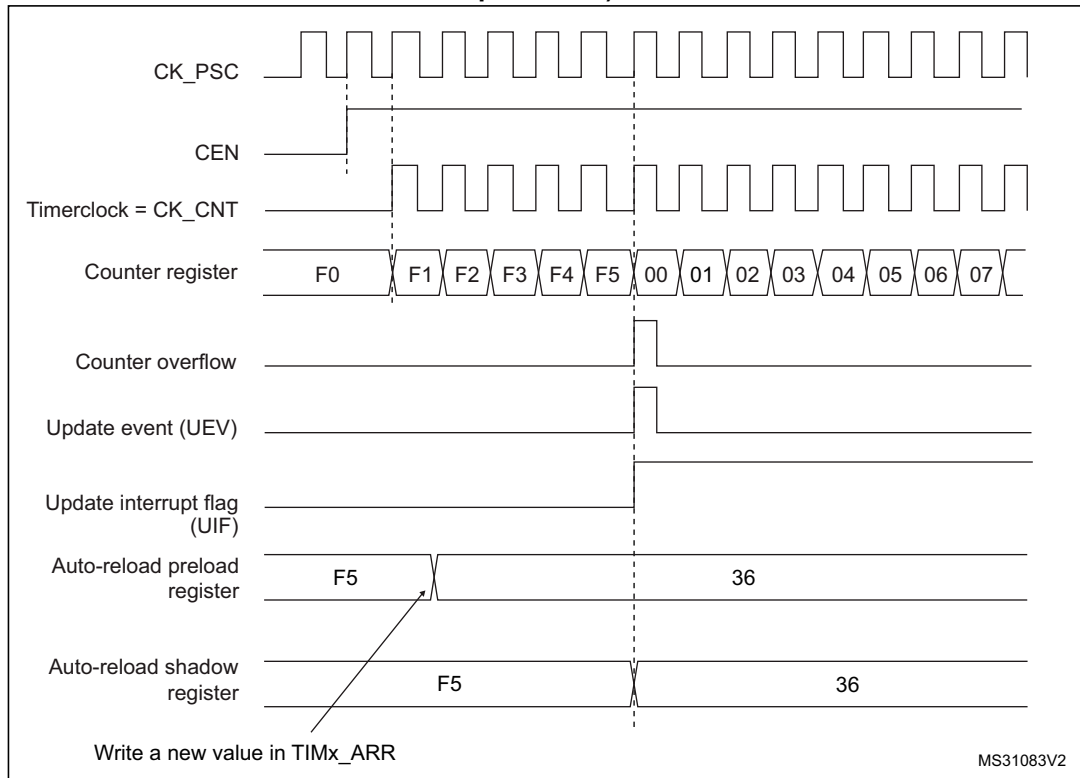


Figure 196. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)

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 197. Counter timing diagram, internal clock divided by 1

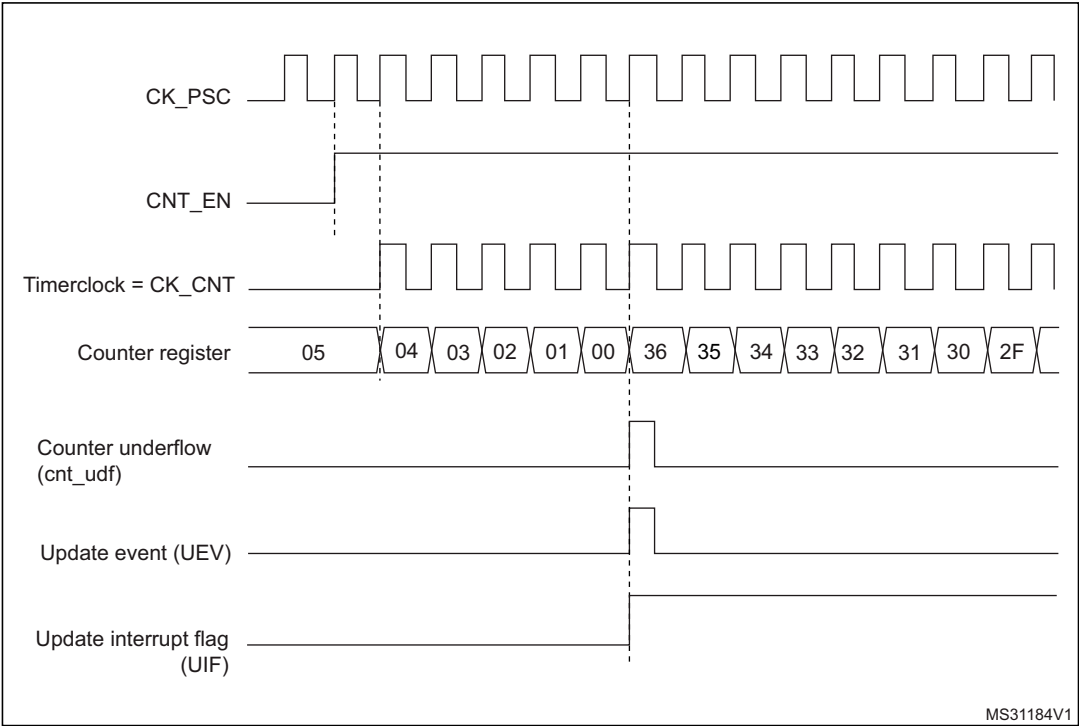


Figure 198. Counter timing diagram, internal clock divided by 2

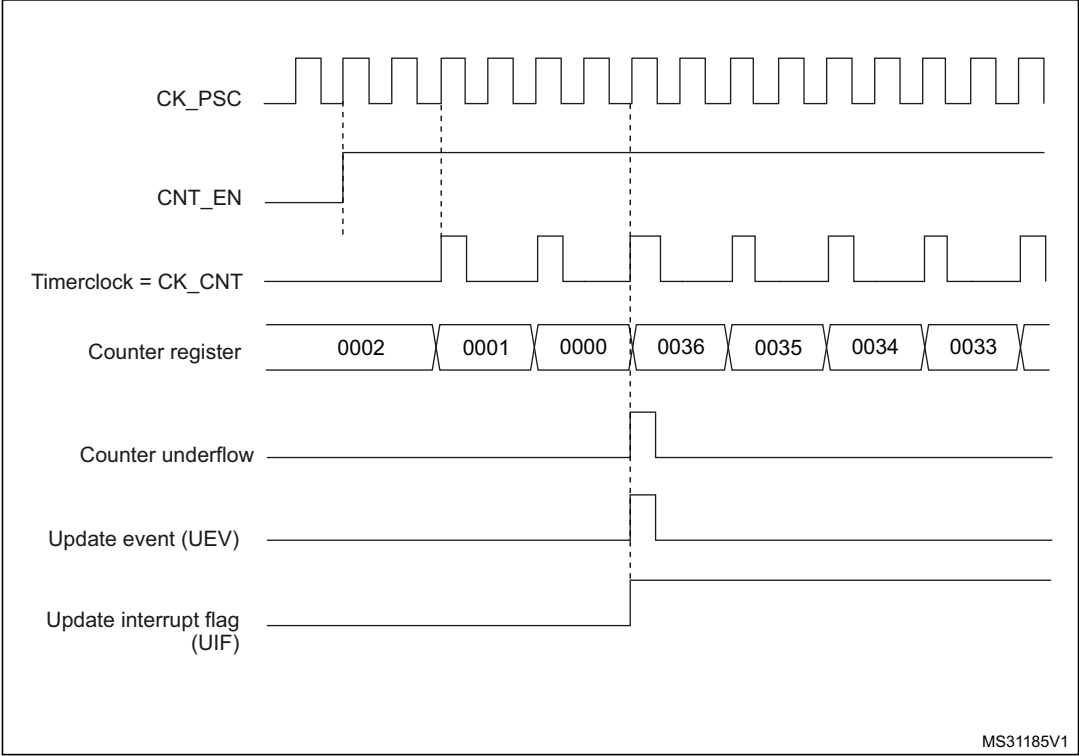


Figure 199. Counter timing diagram, internal clock divided by 4

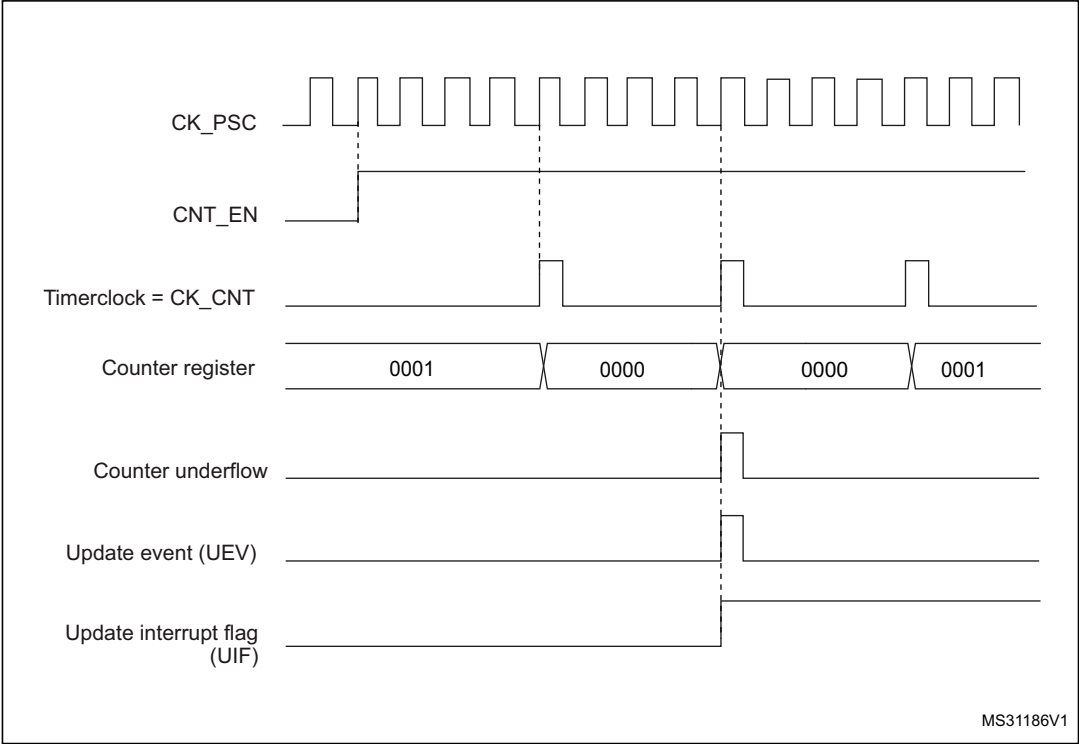


Figure 200. Counter timing diagram, internal clock divided by N

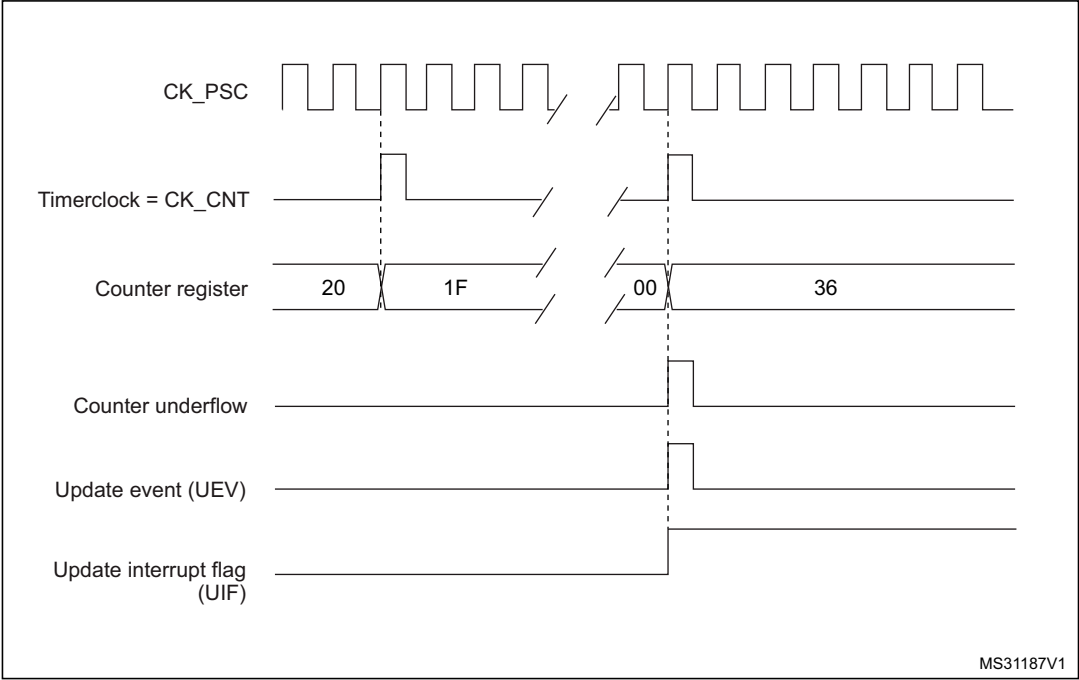
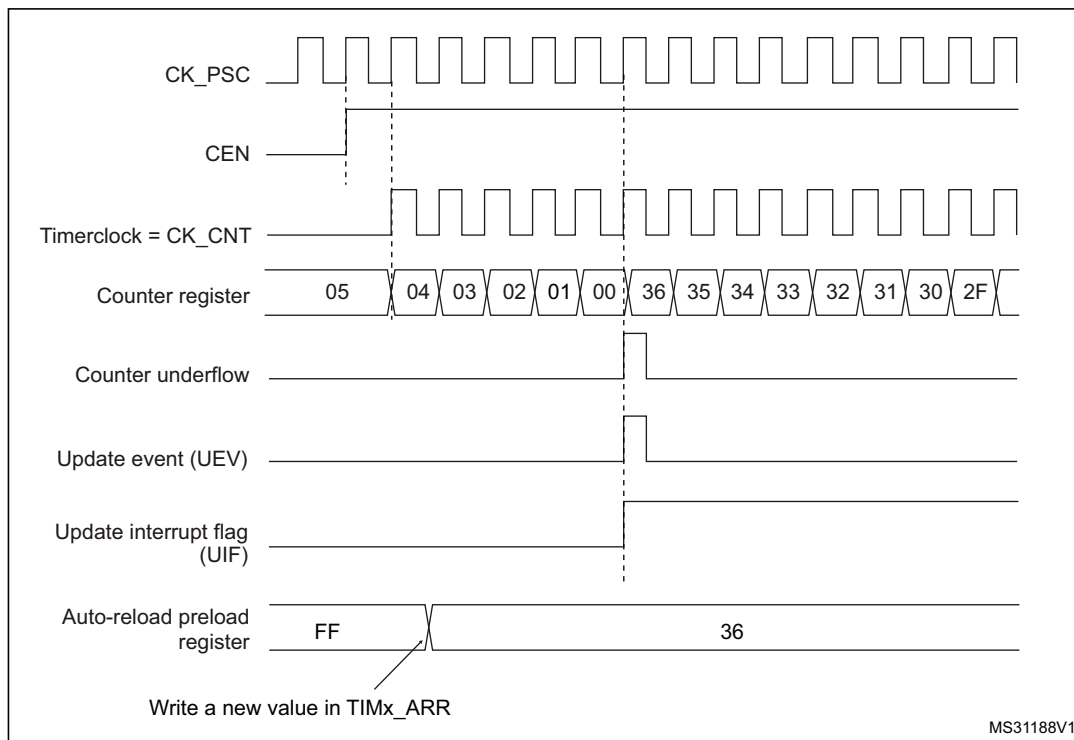


Figure 201. Counter timing diagram, Update event when repetition counter is not used**Center-aligned mode (up/down counting)**

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or

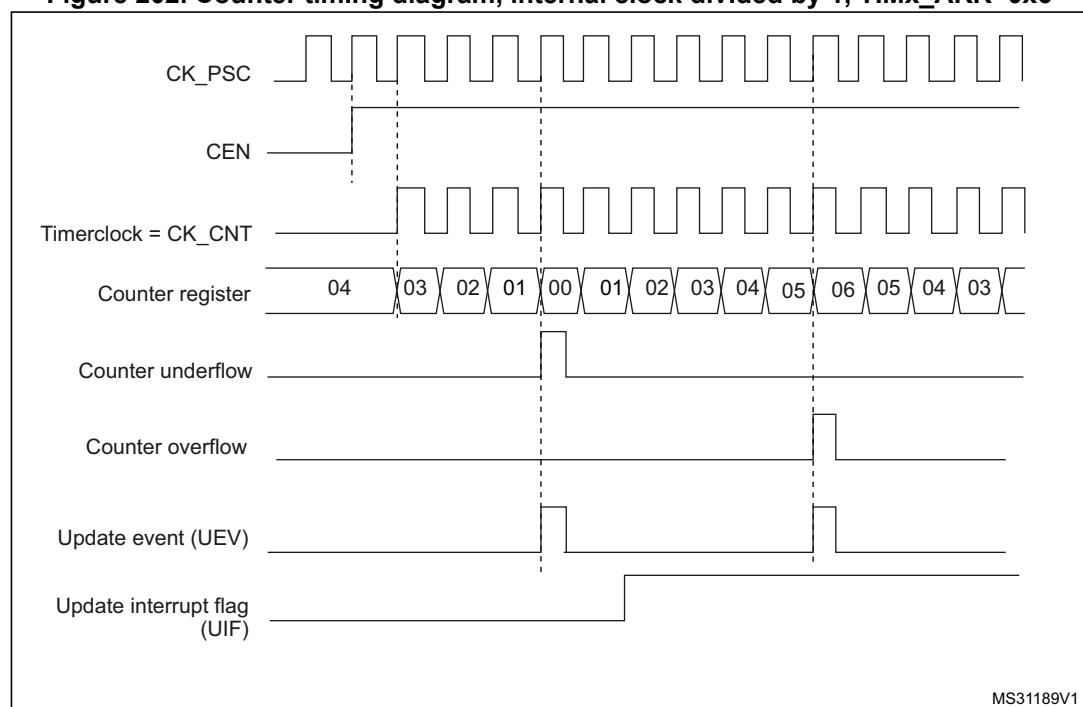
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 202. 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 21.4.1: TIMx control register 1 \(TIMx_CR1\) on page 592](#)).

Figure 203. Counter timing diagram, internal clock divided by 2

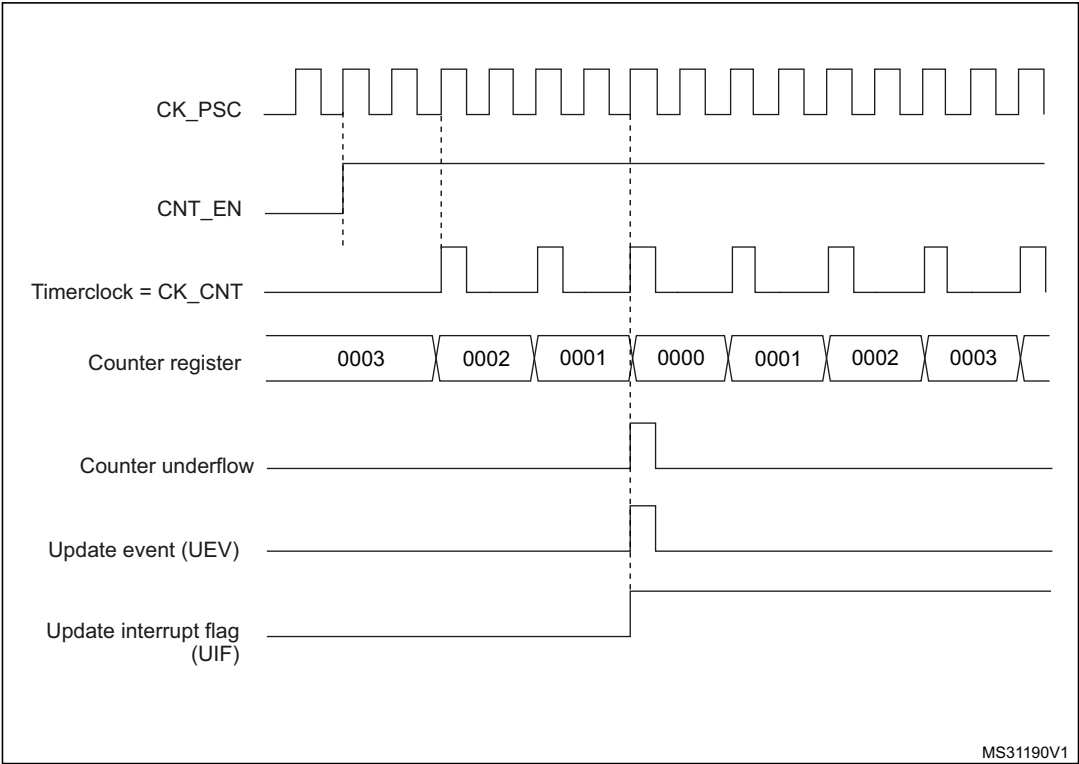
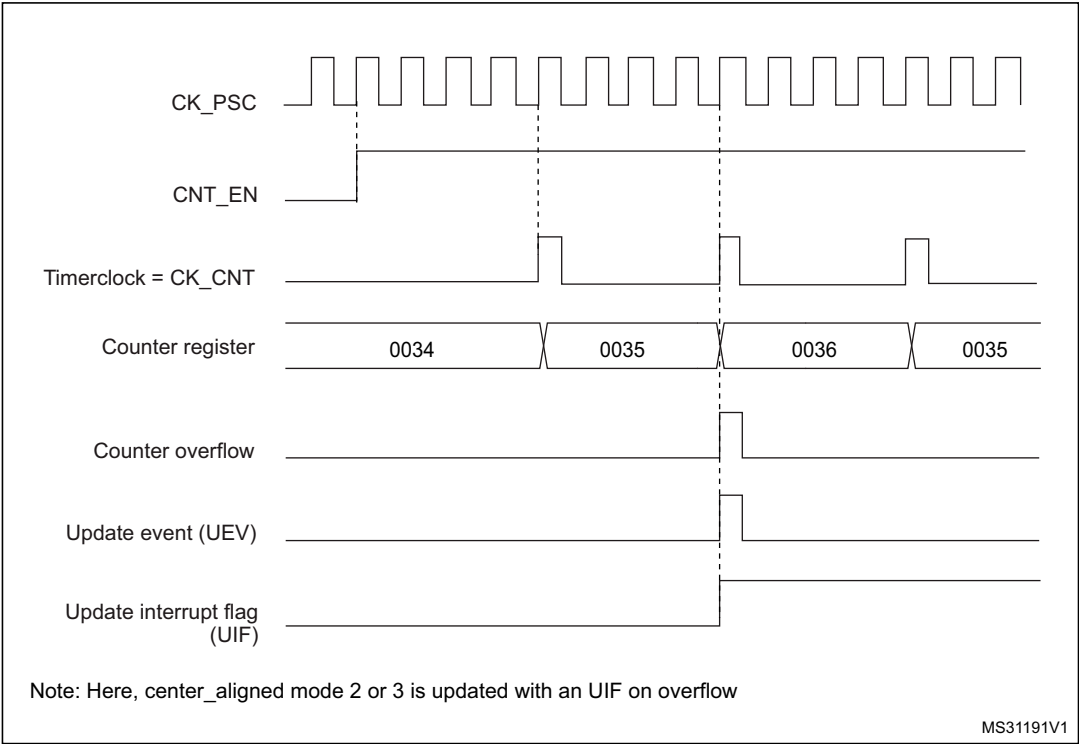


Figure 204. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 205. Counter timing diagram, internal clock divided by N

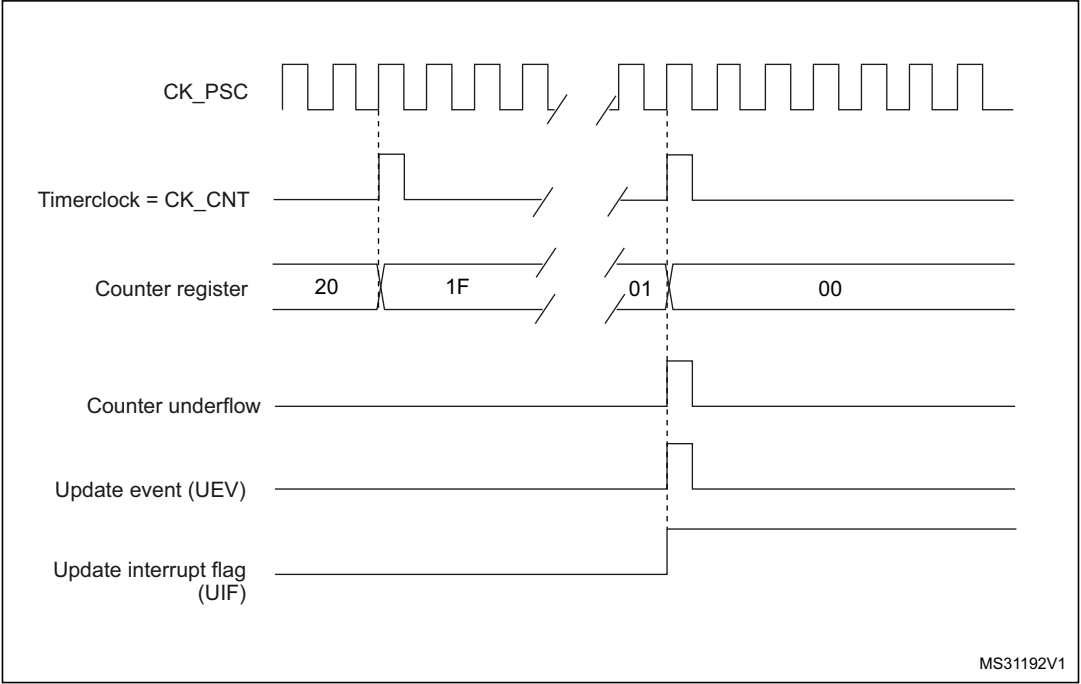


Figure 206. Counter timing diagram, Update event with ARPE=1 (counter underflow)

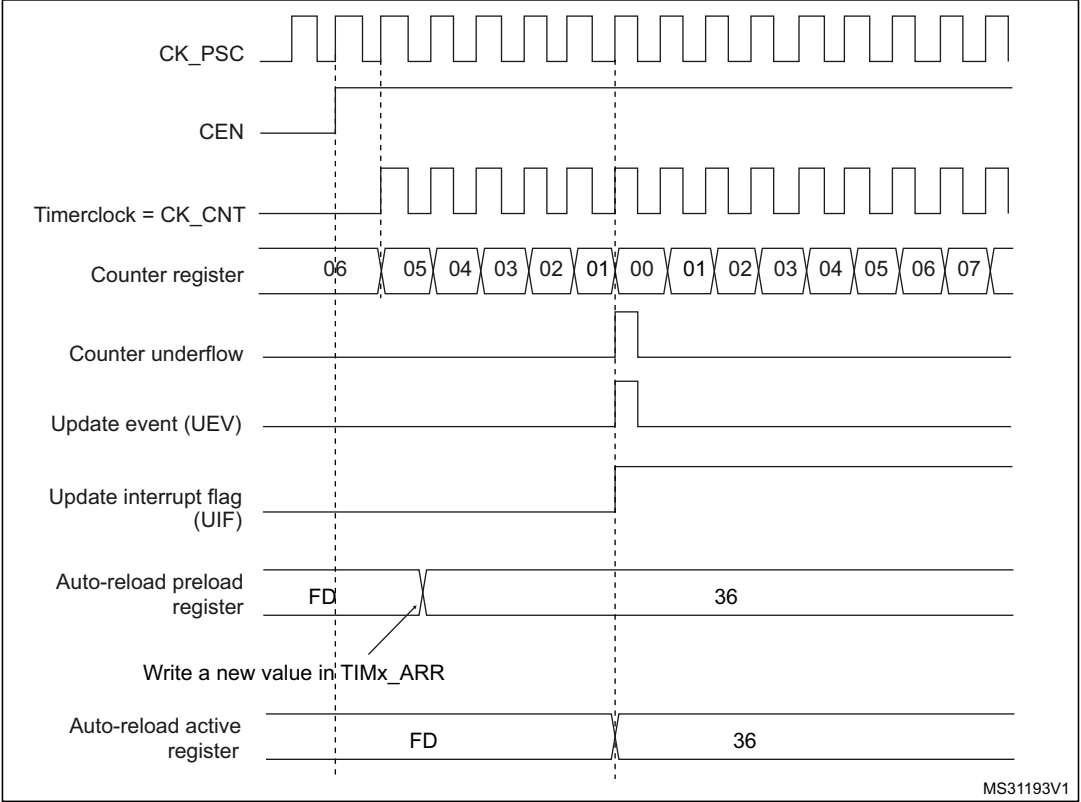
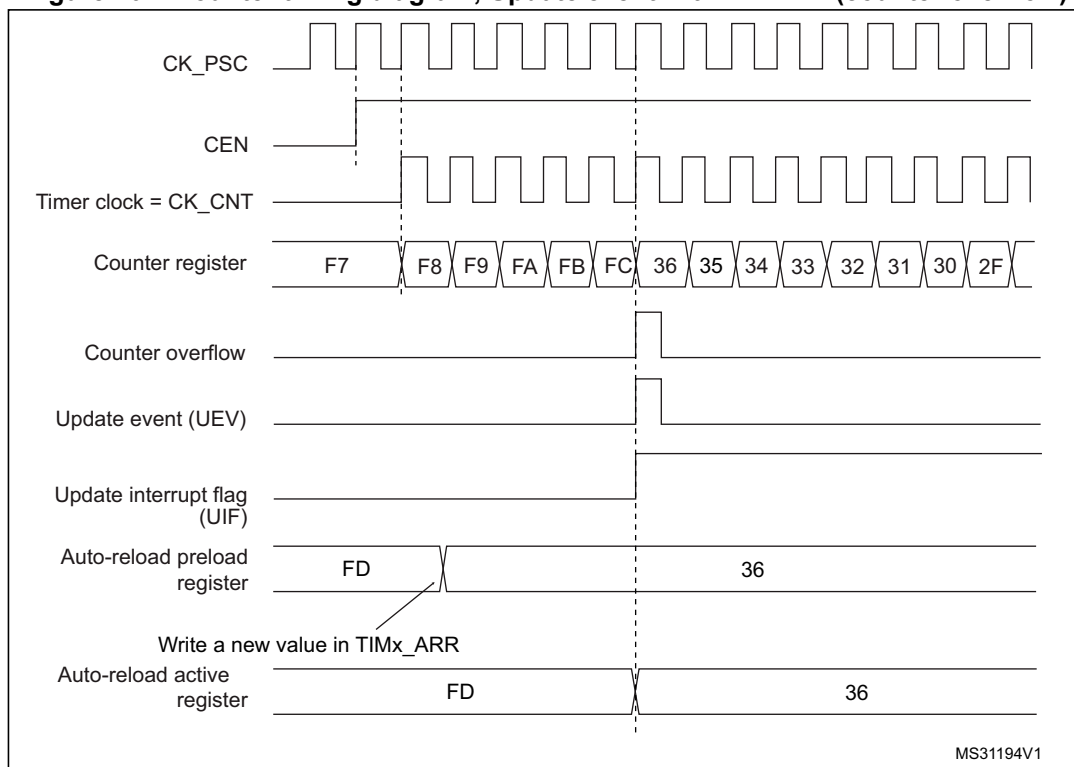


Figure 207. Counter timing diagram, Update event with ARPE=1 (counter overflow)

21.3.3 Clock selection

The counter clock can be provided by the following clock sources:

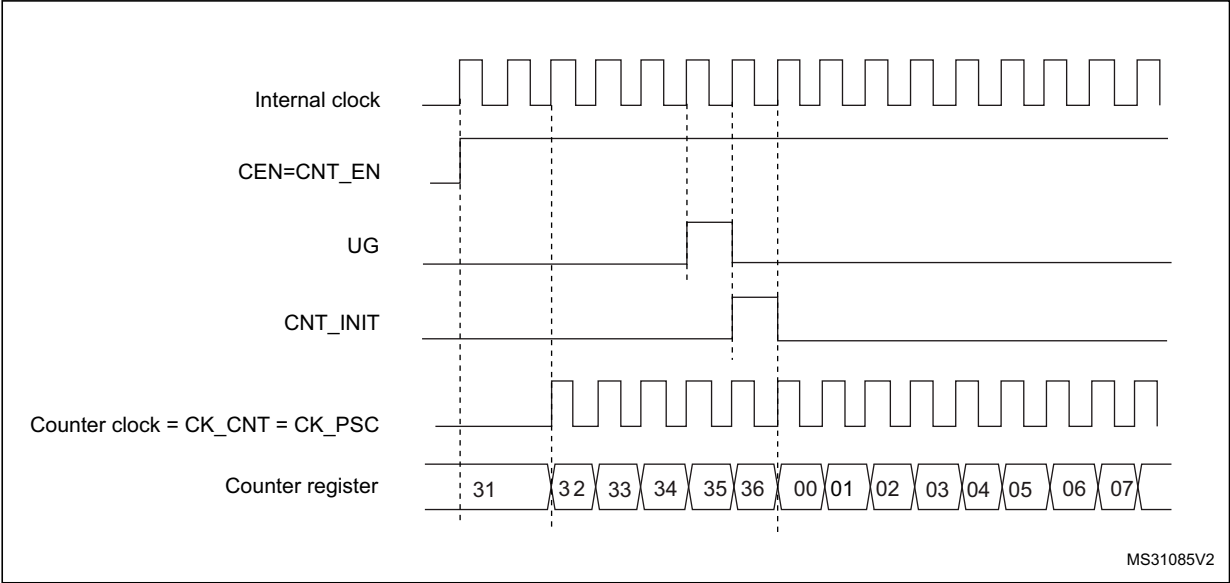
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, you can configure Timer 13 to act as a prescaler for Timer 2. Refer to : [Using one timer as prescaler for another on page 586](#) 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 208 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

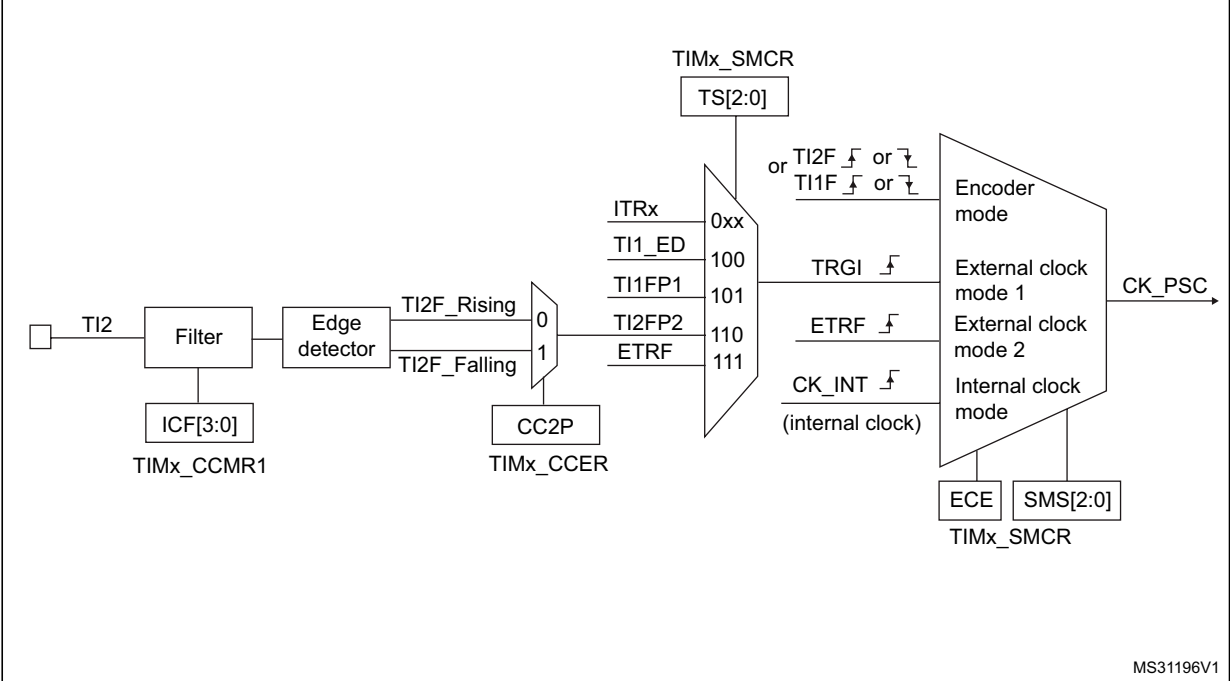
Figure 208. Control circuit in normal mode, internal clock divided by 1



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 209. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= '01 in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).

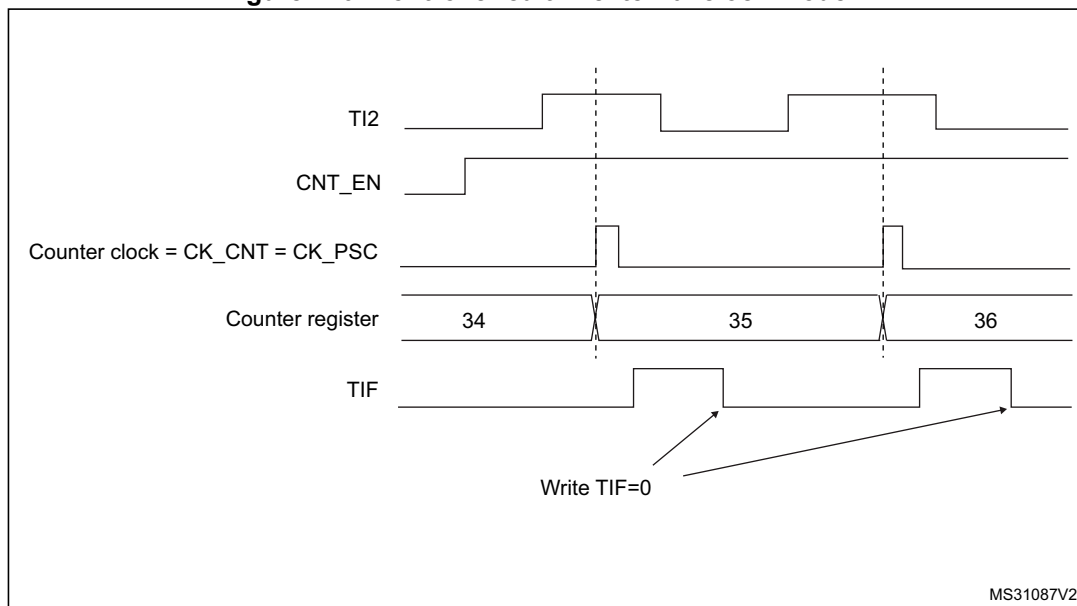
Note: The capture prescaler is not used for triggering, so you don't need to configure it.

3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 and CC2NP=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 210. Control circuit in external clock mode 1



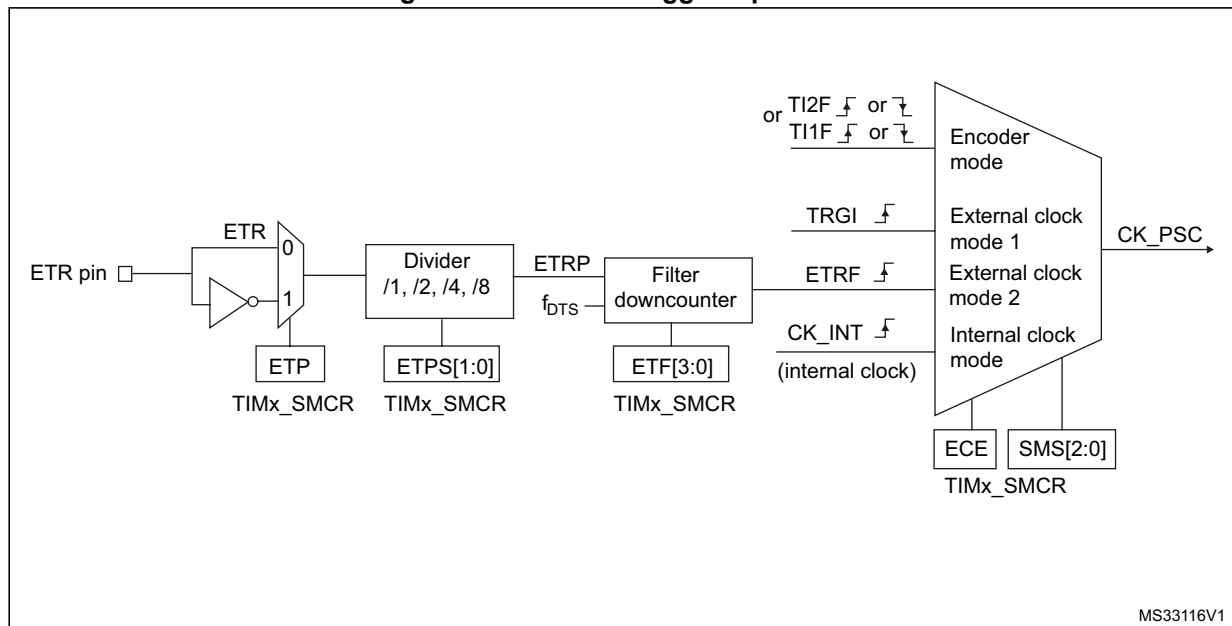
External clock source mode 2

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 211](#) gives an overview of the external trigger input block.

Figure 211. External trigger input block



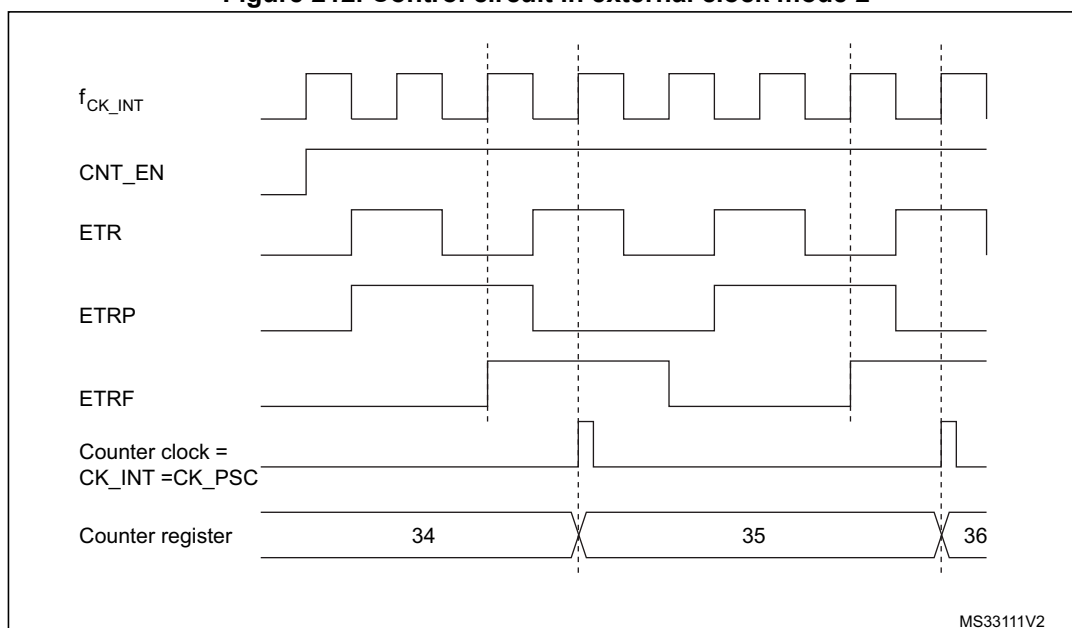
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 212. Control circuit in external clock mode 2



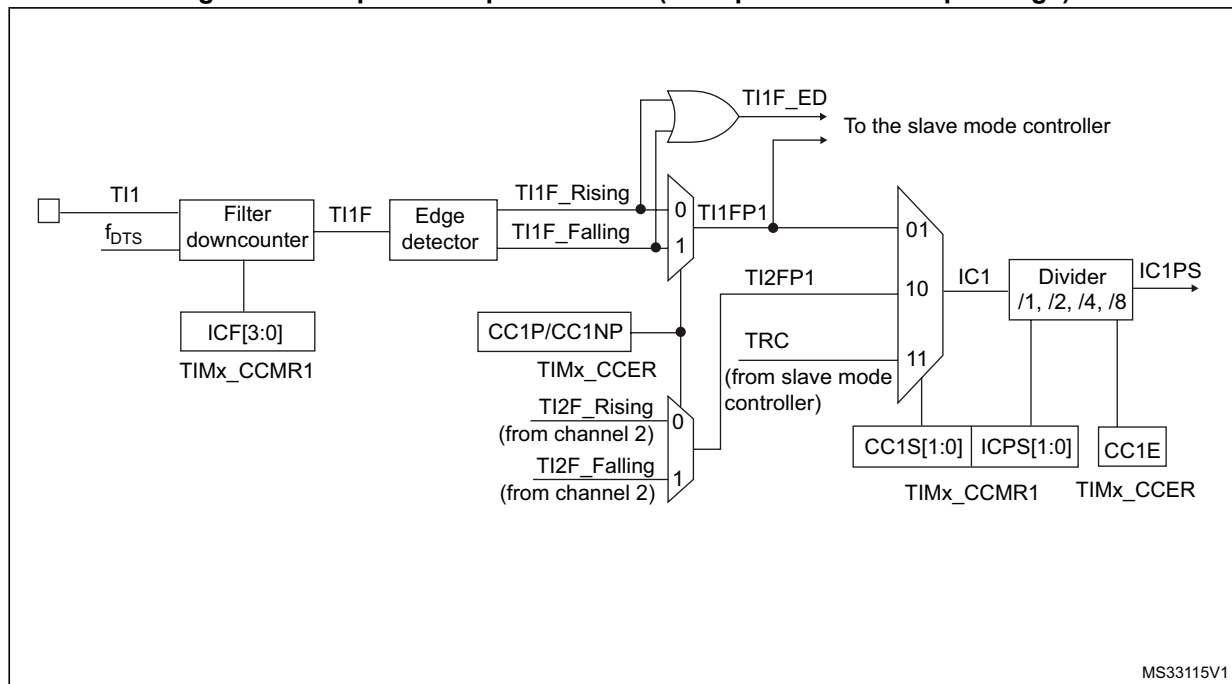
21.3.4 Capture/compare channels

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 213. 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 214. Capture/compare channel 1 main circuit

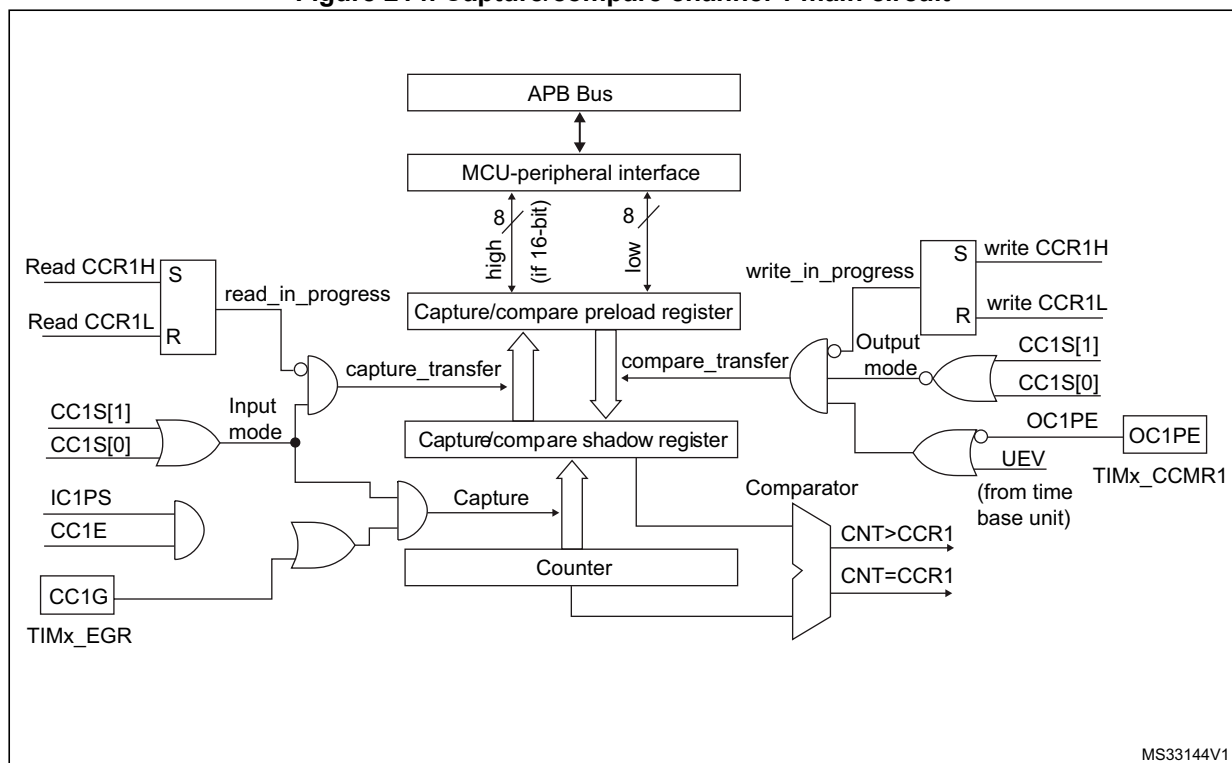
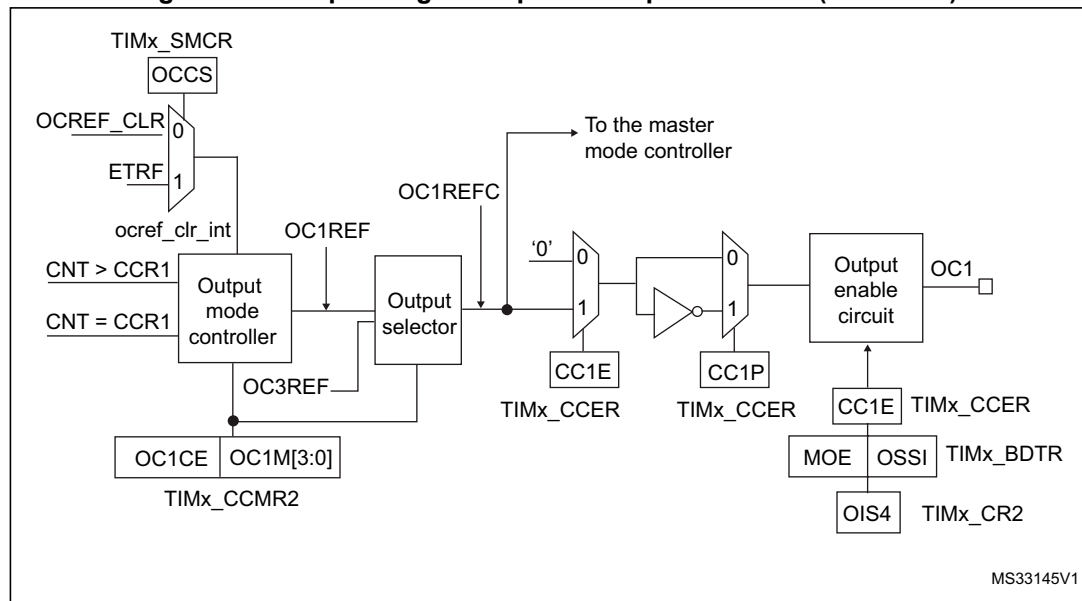


Figure 215. Output stage of capture/compare channel (channel 1)

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

21.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
2. Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

3. Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

21.3.6 PWM input mode

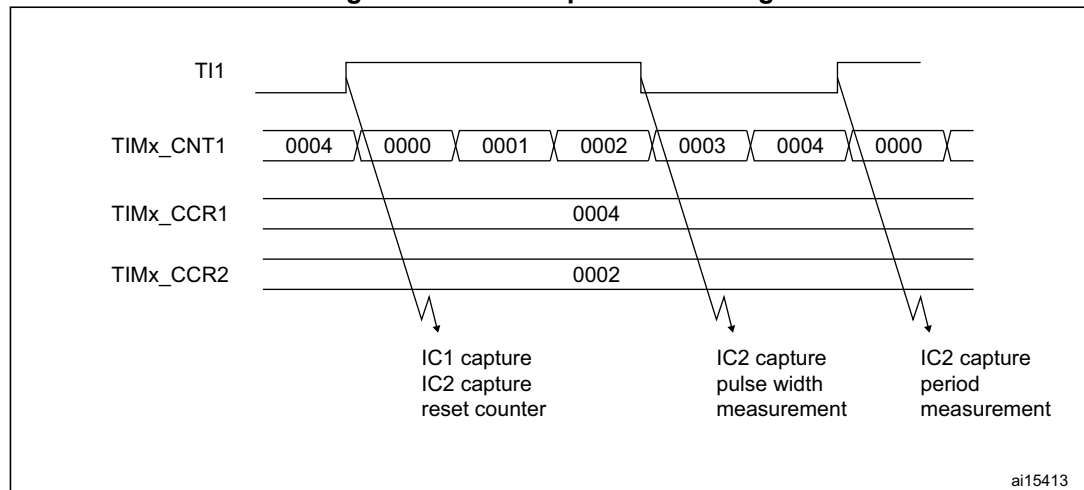
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to '0' and the CC1NP bit to '0' (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to '1' and the CC2NP bit to '0' (active on falling edge).
5. Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 216. 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.

21.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/OCx) to its active level, you just need to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

21.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

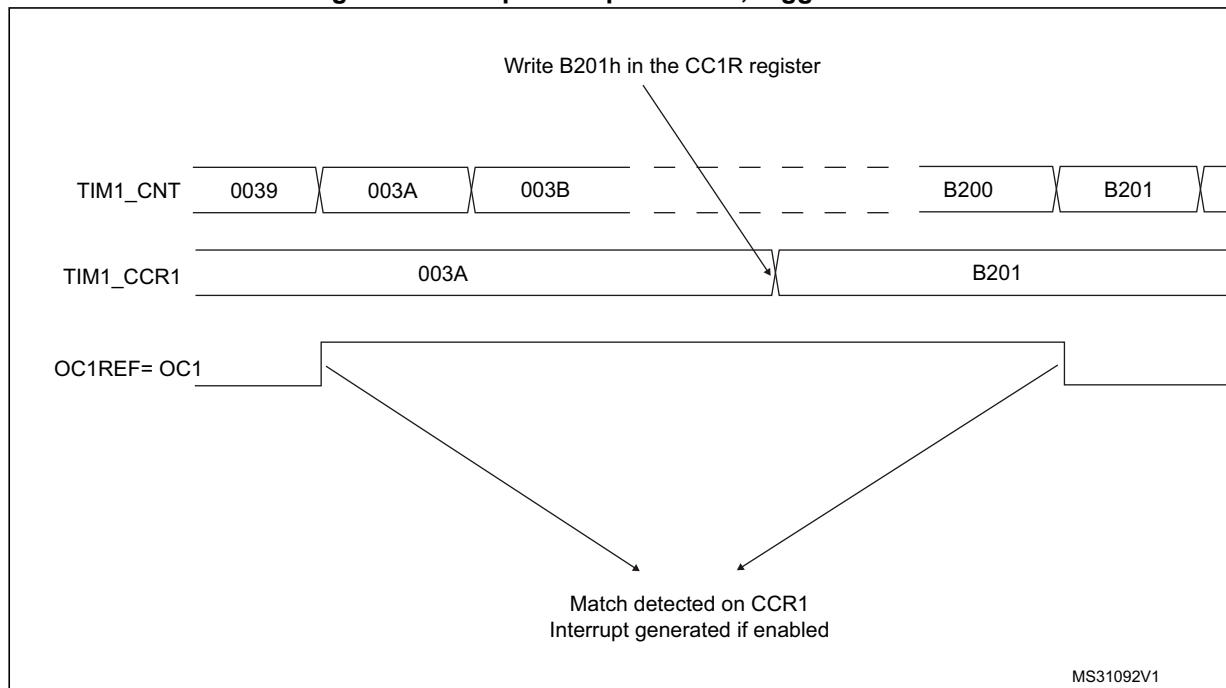
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, you must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 217](#).

Figure 217. Output compare mode, toggle on OC1



21.3.9 PWM mode

Pulse width modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or '111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether $TIMx_CCRx \leq TIMx_CNT$ or $TIMx_CNT \leq TIMx_CCRx$ (depending on the direction of the counter). However, to comply with the OCREF_CLR functionality (OCREF can be

cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison 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.

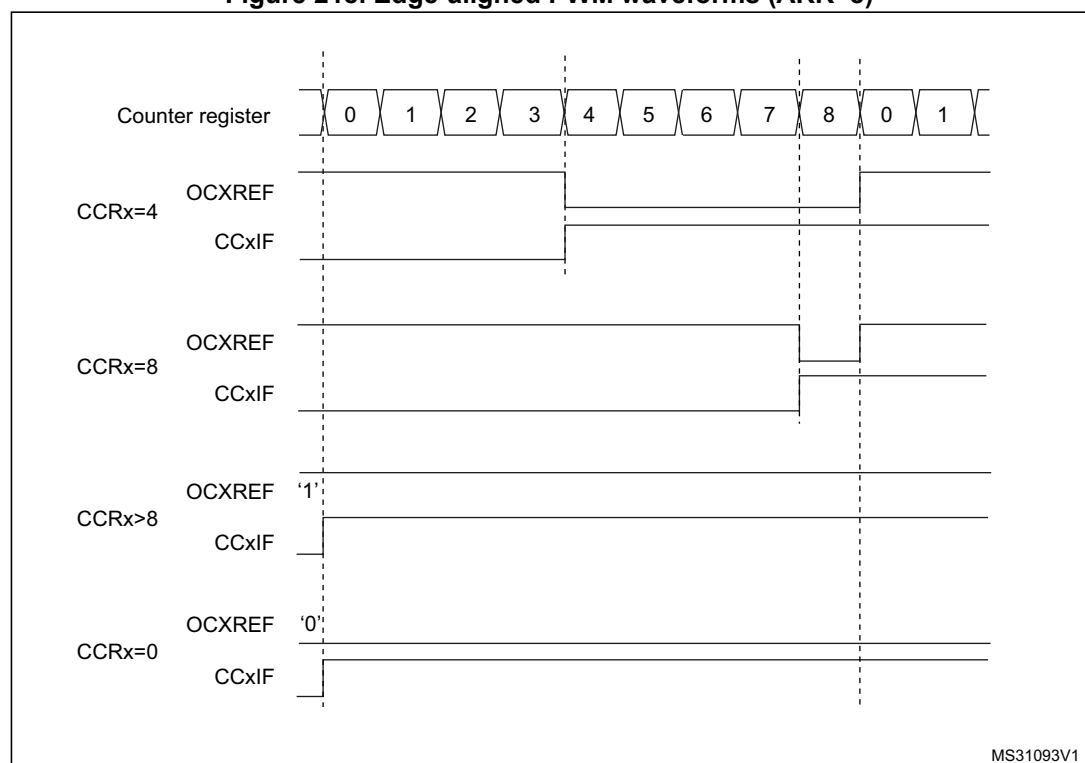
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to [Upcounting mode on page 550](#).

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 218](#) shows some edge-aligned PWM waveforms in an example where $TIMx_ARR=8$.

Figure 218. 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 553](#).

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.

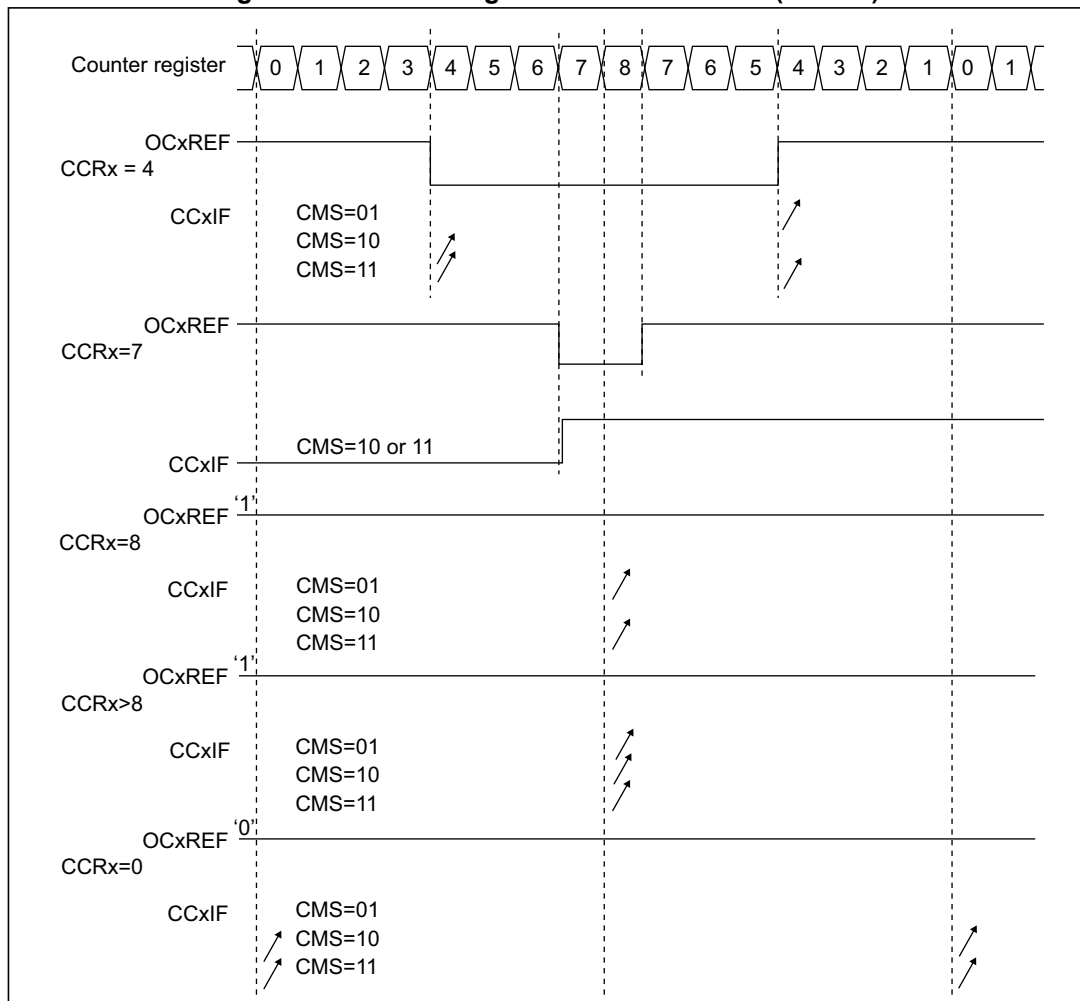
PWM center-aligned mode

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 [Center-aligned mode \(up/down counting\) on page 556](#).

[Figure 219](#) 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 219. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
 - The direction is not updated if you write a value in the counter that is greater than the auto-reload value (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
 - The direction is updated if you write 0 or write the TIMx_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

21.3.10 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

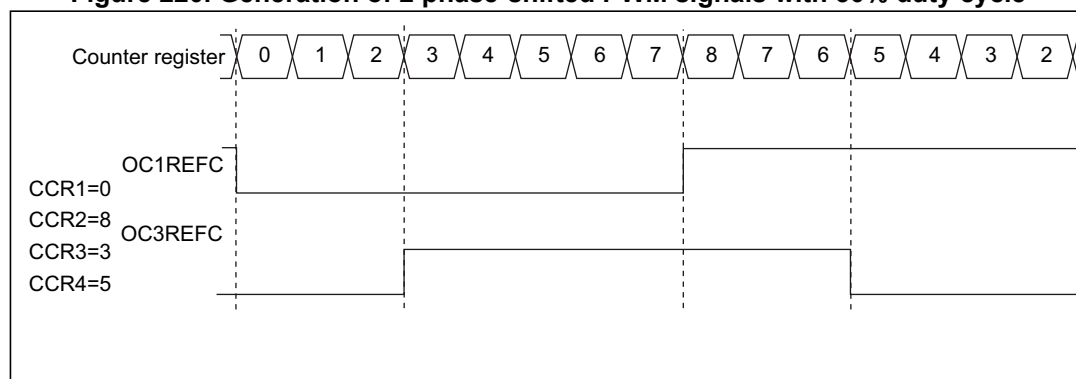
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1110' (Asymmetric PWM mode 1) or '1111' (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

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 220 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 220. Generation of 2 phase-shifted PWM signals with 50% duty cycle



21.3.11 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.

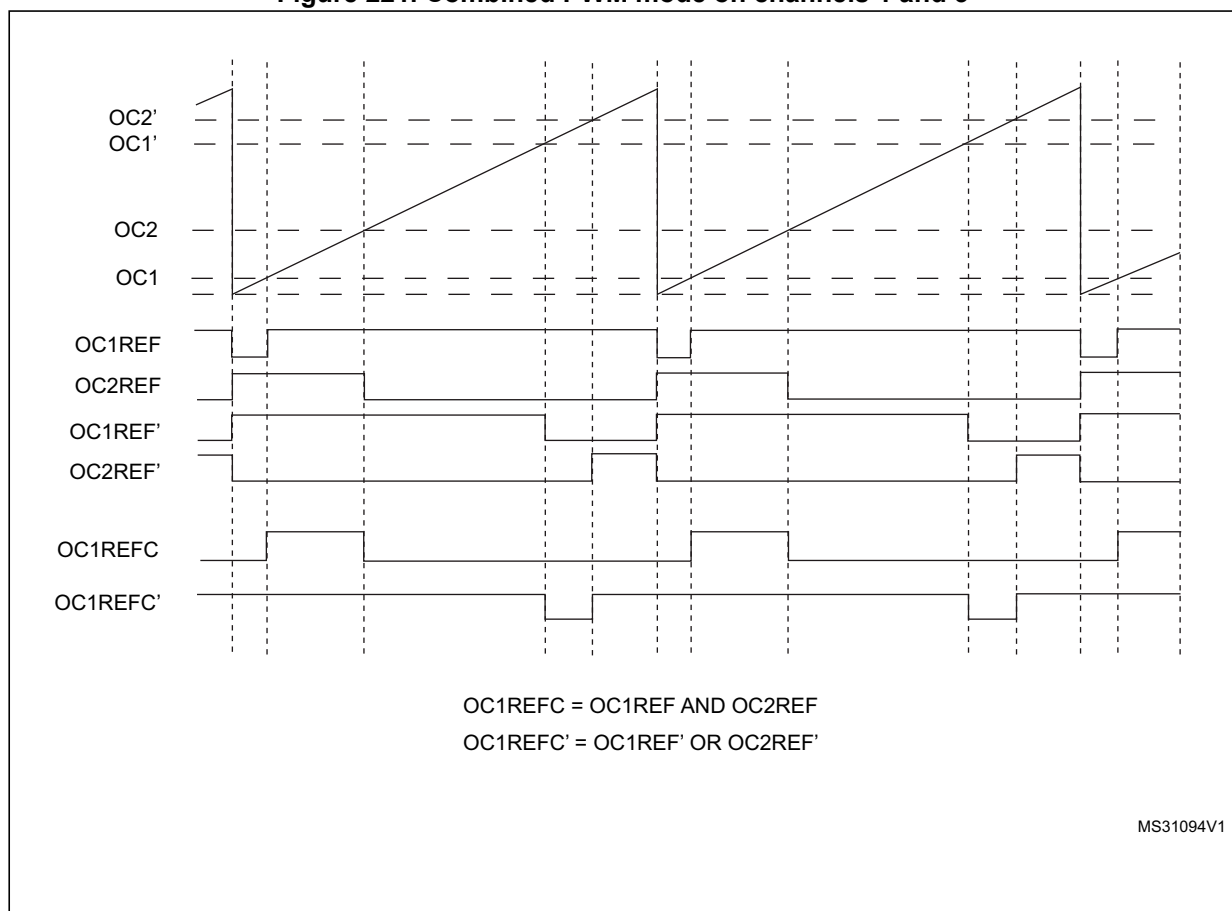
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

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 221 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 221. Combined PWM mode on channels 1 and 3



21.3.12 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_INPUT (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. 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.

When ETRF is chosen, ETR must be configured as follows:

The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx_CCMRx register). OCxREF remains low until the next update event (UEV) occurs.

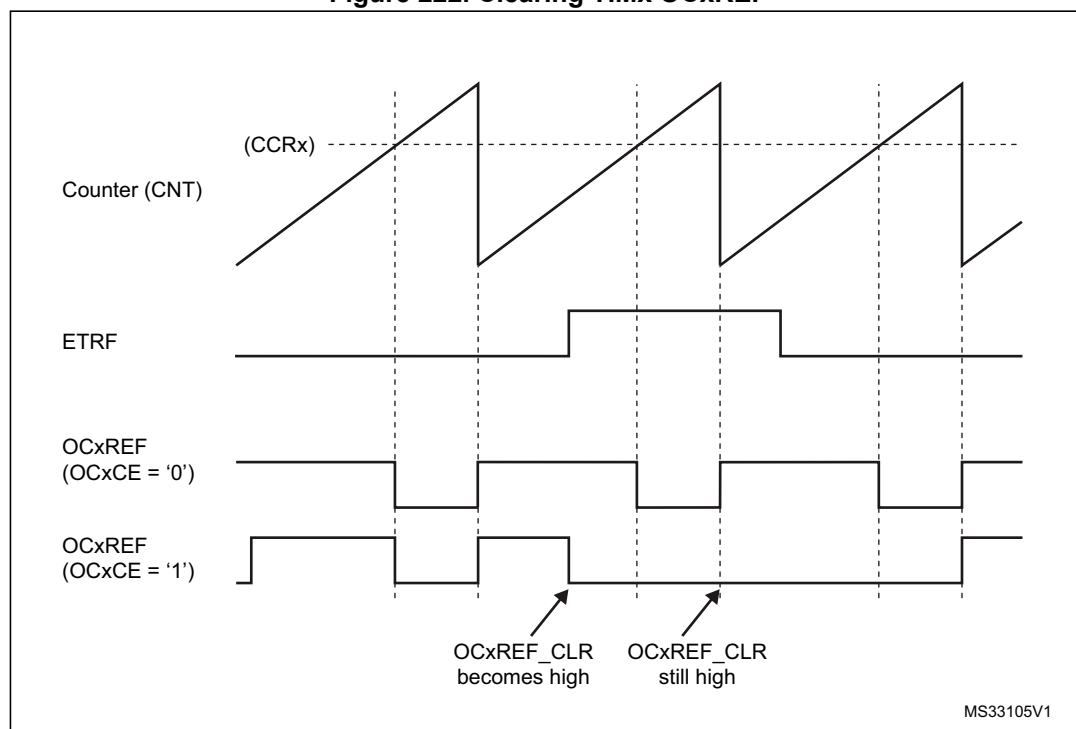
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 222 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 222. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if $CCR_x > ARR$), OCxREF is enabled again at the next counter overflow.

21.3.13 One-pulse mode

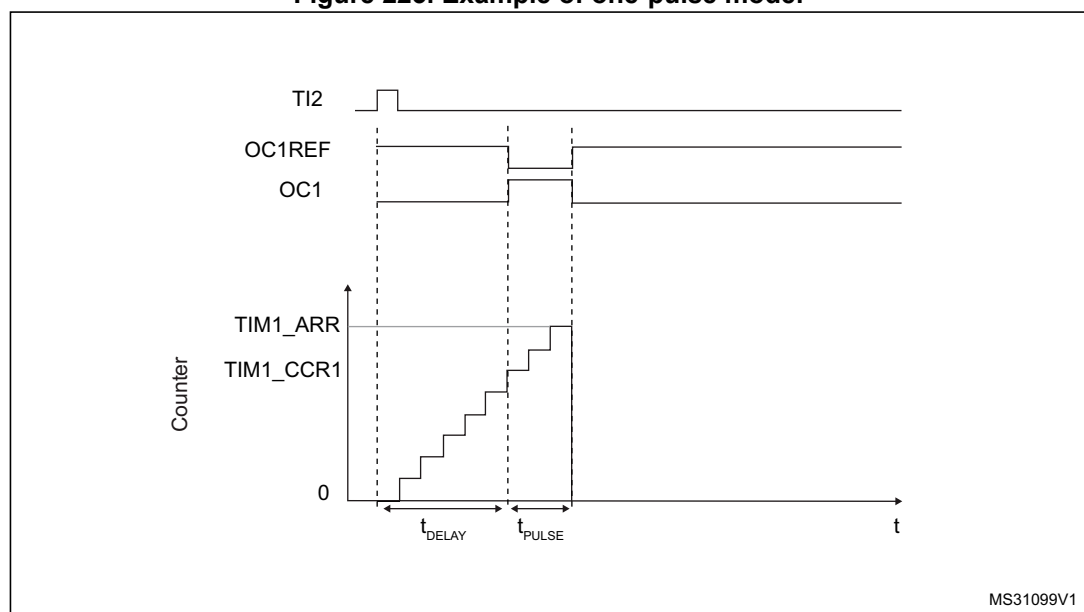
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $CNT < CCRx \leq ARR$ (in particular, $0 < CCRx$),

Figure 223. Example of one-pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 on TI2 by writing IC2S=01 in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=110 in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

You only want 1 pulse (Single mode), so you write '1' in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

21.3.14 Retriggerable one pulse mode (OPM)

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 21.3.13](#):

- 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.

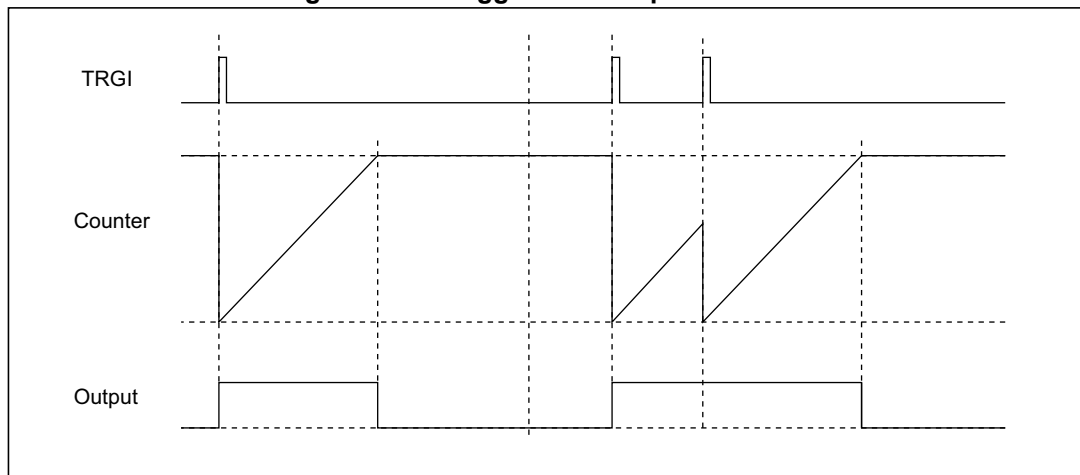
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

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 224 Retriggerable one pulse mode



21.3.15 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, you can program the input filter 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 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 you must configure TIMx_ARR before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the-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 don't 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

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 225 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = '0' (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = '0' (TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)

Figure 225. Example of counter operation in encoder interface mode

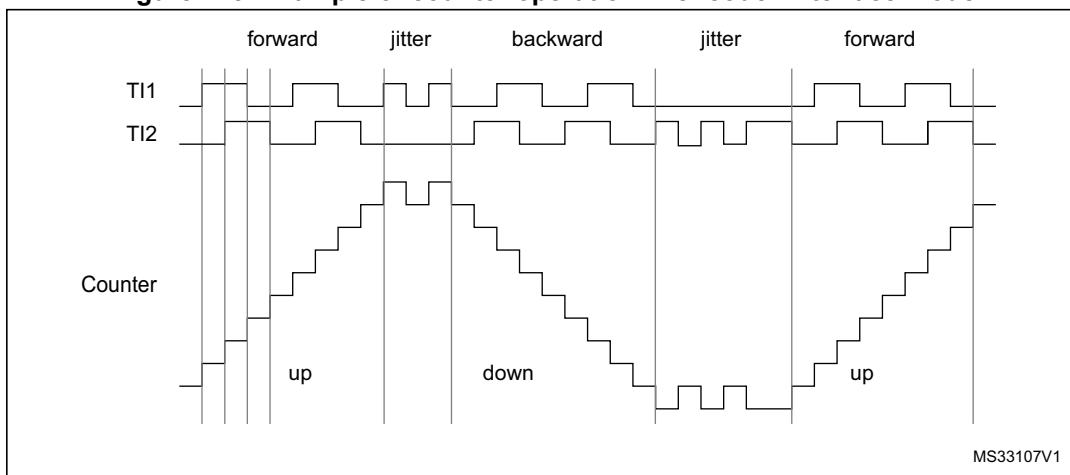
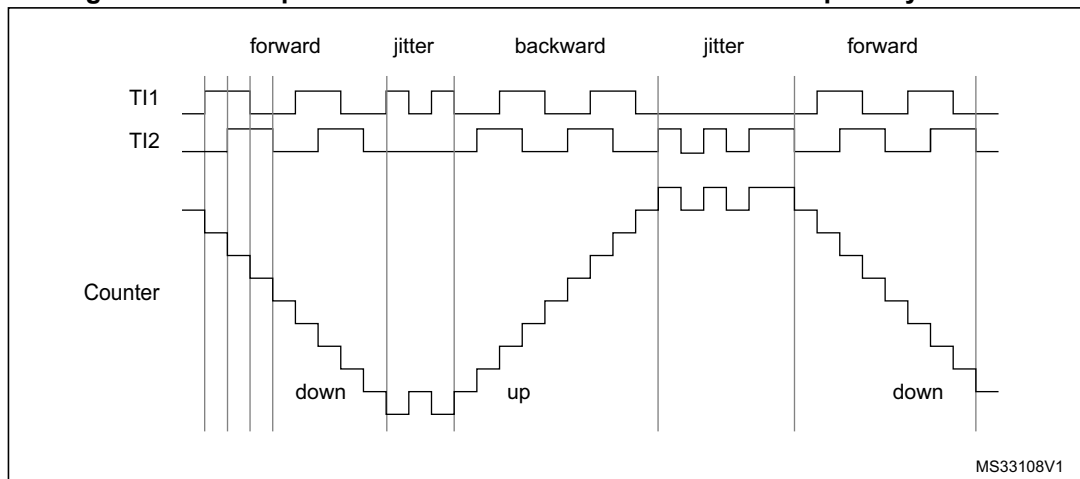


Figure 226 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 226. Example of encoder interface mode with TI1FP1 polarity inverted

The timer, when configured in Encoder Interface mode provides information on the sensor's current position. You can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. You can do this by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

21.3.16 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 allows 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).

21.3.17 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 20.3.23: Interfacing with Hall sensors on page 502](#).

21.3.18 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.

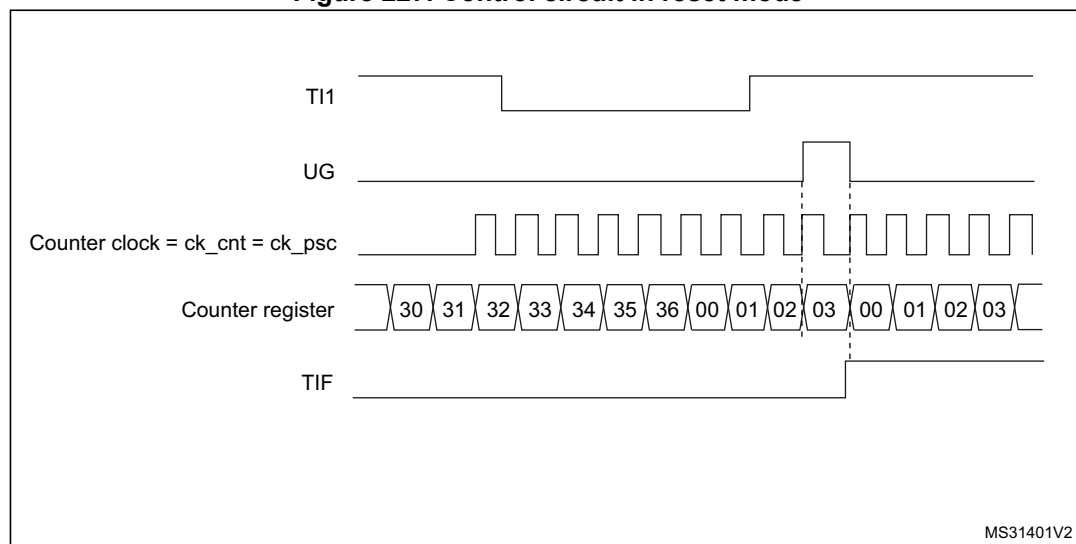
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 227. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

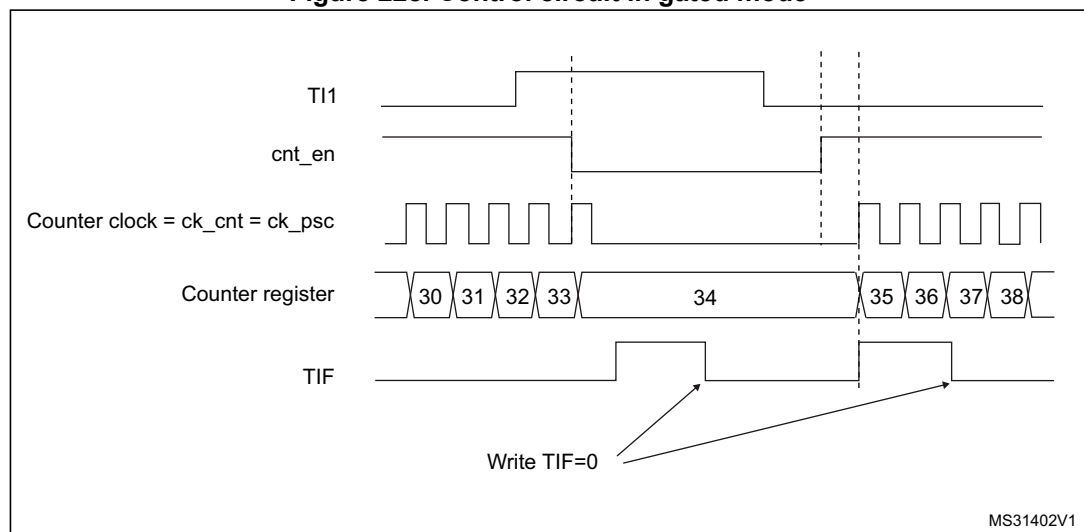
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 228. 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 don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write

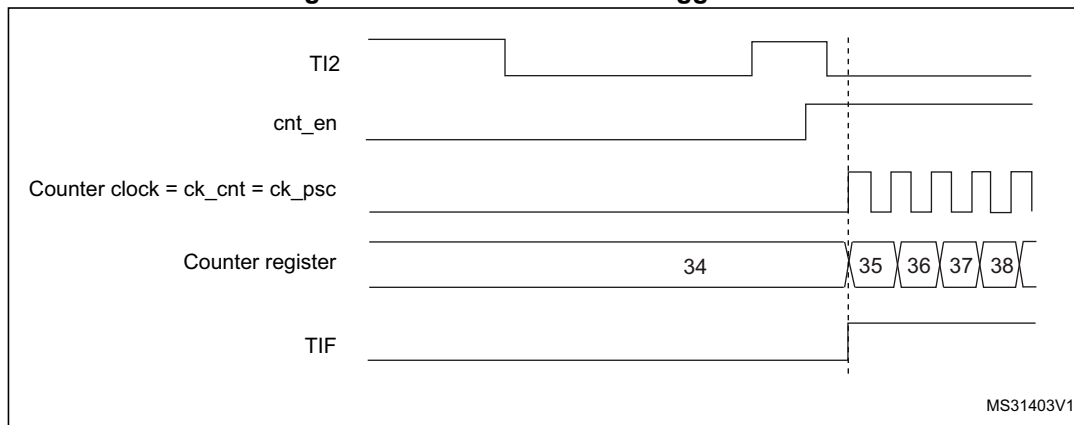
CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 229. 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.

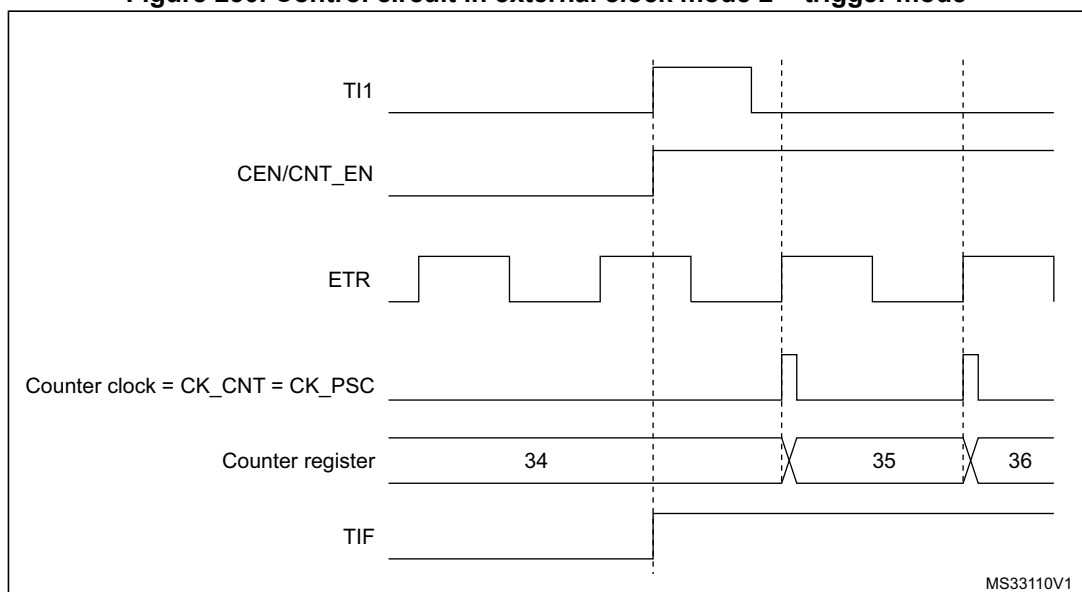
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

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=101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 230. Control circuit in external clock mode 2 + trigger mode



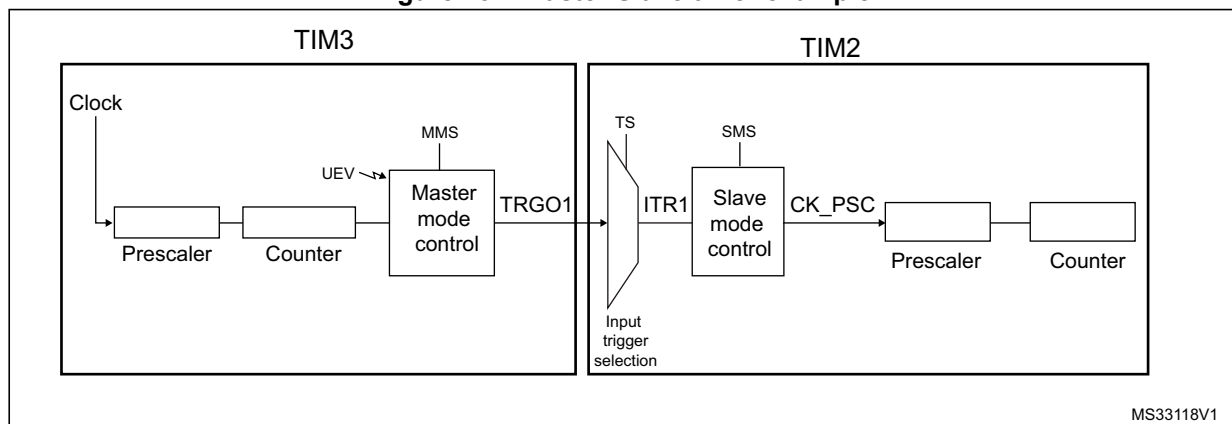
21.3.19 Timer synchronization

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 231: Master/Slave timer example presents an overview of the trigger selection and the master mode selection blocks.

Using one timer as prescaler for another

Figure 231. Master/Slave timer example



For example, you can configure TIM3 to act as a prescaler for TIM2. Refer to [Figure 231](#). To do this:

1. Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If you write MMS=010 in the TIM3_CR2 register, a rising edge is output on TRGO1 each time an update event is generated.
2. To connect the TRGO1 output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR12 as internal trigger. You select this through the TS bits in the TIM2_SMCR register (writing TS=000).
3. Then you put the slave mode controller in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM3 trigger signal (which correspond to the TIM3 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

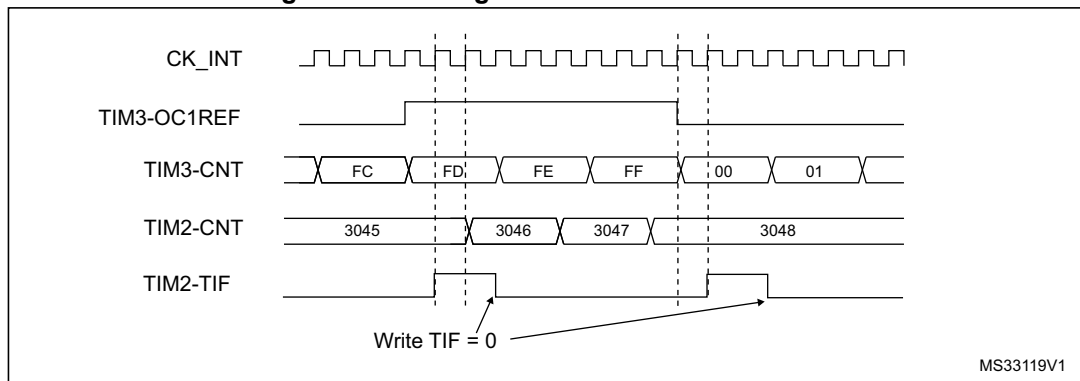
Note: If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

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 231](#) 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=000 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 232. Gating TIM2 with OC1REF of TIM3

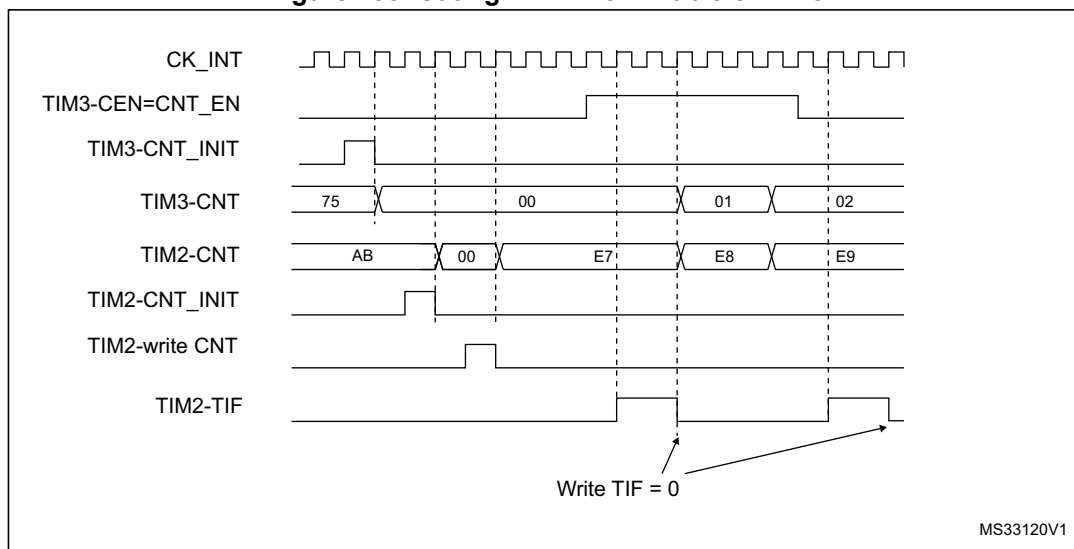
MS33119V1

In the example in [Figure 232](#), the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. You can then write any value you want in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example, we synchronize TIM3 and TIM2. TIM3 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM3 is disabled by writing '0' to the CEN bit in the TIM3_CR1 register:

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=000 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 233. Gating TIM2 with Enable of TIM3

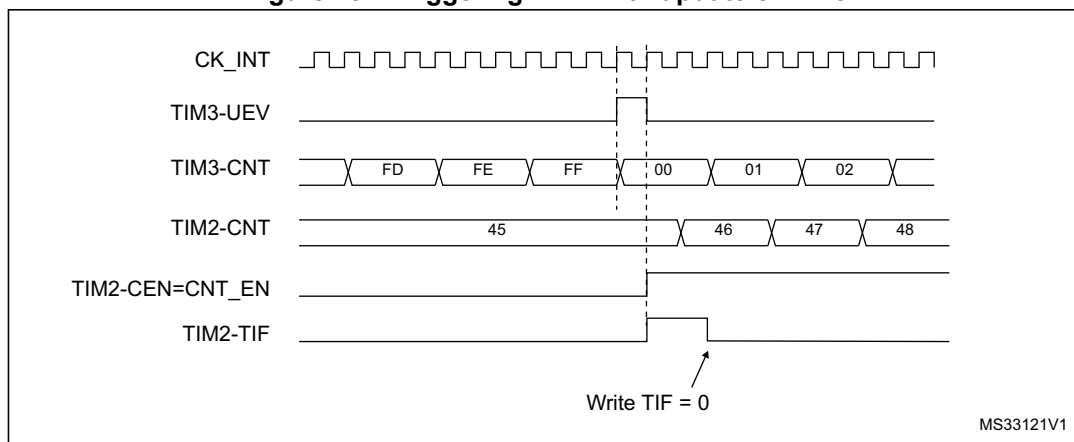


Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 3. Refer to [Figure 231](#) 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 ($f_{CK_CNT} = f_{CK_INT}/3$).

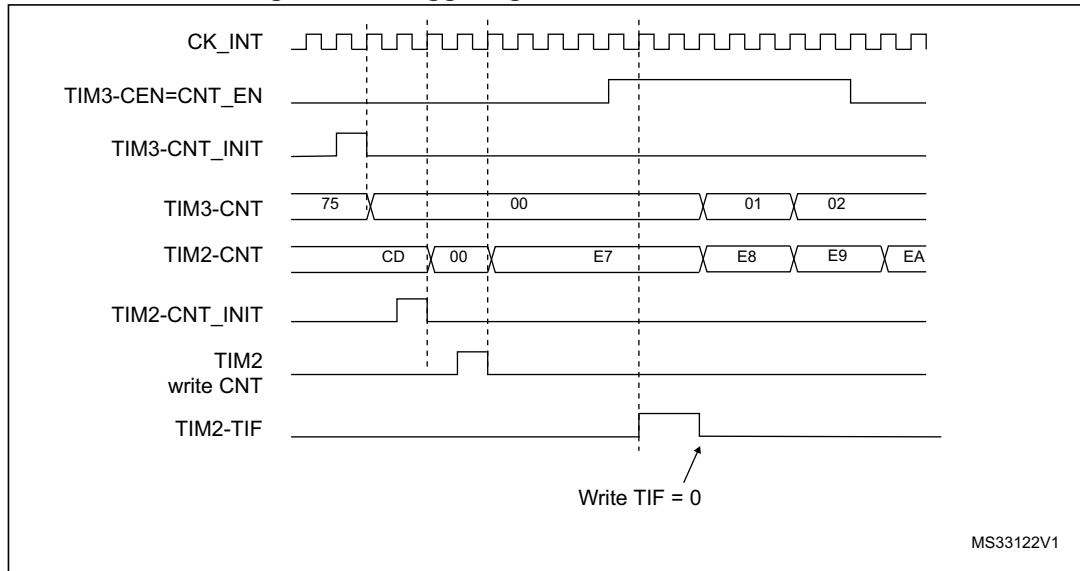
1. Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register).
2. Configure the TIM3 period (TIM3_ARR registers).
3. Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM3 by writing '1 in the CEN bit (TIM3_CR1 register).

Figure 234. Triggering TIM2 with update of TIM3



As in the previous example, you can initialize both counters before starting counting. [Figure 235](#) shows the behavior with the same configuration as in [Figure 234](#) but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 235. Triggering TIM2 with Enable of TIM3



Using one timer as prescaler for another timer

For example, you can configure TIM3 to act as a prescaler for TIM2. Refer to [Figure 231](#) for connections. To do this:

1. Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register). then it outputs a periodic signal on each counter overflow.
2. Configure the TIM3 period (TIM3_ARR registers).
3. Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
4. Configure TIM2 in external clock mode 1 (SMS=111 in TIM2_SMCR register).
5. Start TIM2 by writing '1 in the CEN bit (TIM2_CR1 register).
6. Start TIM3 by writing '1 in the CEN bit (TIM3_CR1 register).

Starting 2 timers synchronously in response to an external trigger

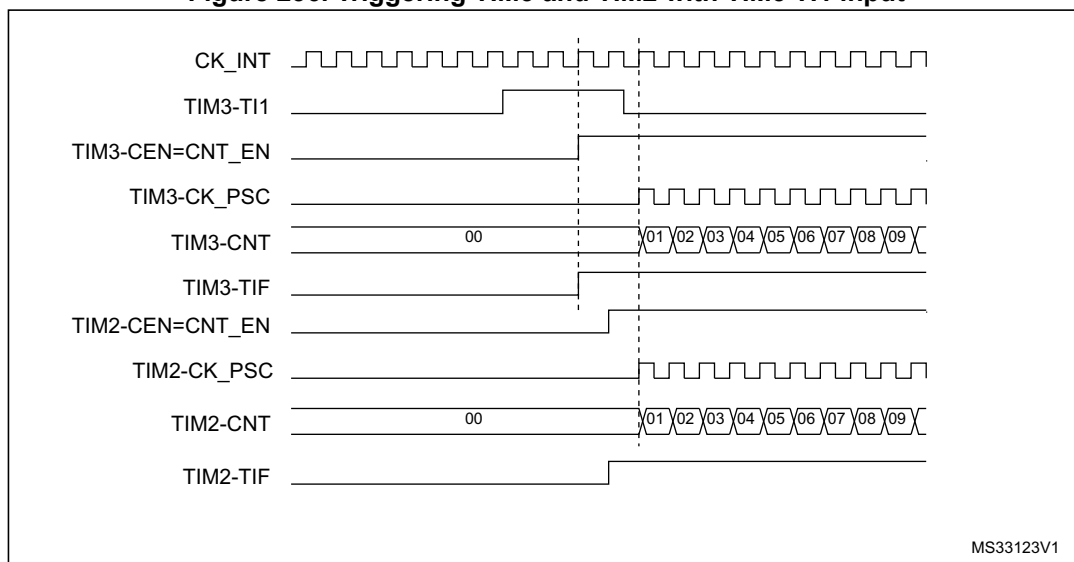
In this example, we set the enable of TIM3 when its TI1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to [Figure 231](#) for connections. To ensure the counters are aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

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=100 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=000 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 you can easily insert an offset between them by writing any of the counter registers (TIMx_CNT). You can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM3.*

Figure 236. Triggering TIM3 and TIM2 with TIM3 TI1 input



21.3.20 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 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.

21.3.21 Debug mode

When the microcontroller enters debug mode (Cortex-M4[®]F core - halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBGMCU module. For more details, refer to [Section 33.14.2: Debug support for timers, watchdog, bxCAN and I2C](#).

21.4 TIM2/TIM3/TIM4 registers

Refer to [Section 2.1](#) 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 TIMx control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIF RE- MAP	Res.	CKD[1:0]		ARPE	CMS		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**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, Tlx),

00: $t_{DTS} = t_{CK_INT}$

01: $t_{DTS} = 2 \times t_{CK_INT}$

10: $t_{DTS} = 4 \times t_{CK_INT}$

11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered

1: TIMx_ARR register is buffered

Bits 6:5 **CMS**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled.

These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

21.4.2 TIMx control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	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 20.3.23: Interfacing with Hall sensors on page 502](#)

Bits 6:4 **MMS**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)

100: **Compare** - OC1REF signal is used as trigger output (TRGO)

101: **Compare** - OC2REF signal is used as trigger output (TRGO)

110: **Compare** - OC3REF signal is used as trigger output (TRGO)

111: **Compare** - OC4REF signal is used as trigger output (TRGO)

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

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.

21.4.3 TIMx slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]
															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 31:17 Reserved, must be kept at reset value.

Bit 16 **SMS[3]**: Slave mode selection - bit 3

Refer to SMS description - bits 2:0

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{\text{ETR}}$ is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge

1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).

2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).

3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[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 events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM**: Master/Slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

- 000: Internal Trigger 0 (ITR0). reserved
- 001: Internal Trigger 1 (ITR1).
- 010: Internal Trigger 2 (ITR2).
- 011: Internal Trigger 3 (ITR3). reserved
- 100: TI1 Edge Detector (TI1F_ED)
- 101: Filtered Timer Input 1 (TI1FP1)
- 110: Filtered Timer Input 2 (TI2FP2)
- 111: External Trigger input (ETRF)

See [Table 119: TIMx internal trigger connection on page 597](#) for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source

- 0: OCREF_CLR_INT is connected to the OCREF_CLR input
- 1: OCREF_CLR_INT is connected to ETRF

Bits 2:0 **SMS**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

- 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 TI2FP2 edge depending on TI1FP1 level.
- 0010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 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=100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Table 119. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1	Reserved	TIM3	TIM4
TIM3	TIM1	TIM2	TIM15	TIM4
TIM4	TIM1	TIM2	TIM3	Reserved

21.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
 0: Trigger DMA request disabled.
 1: Trigger DMA request enabled.

Bit 13 Reserved, 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.

21.4.5 TIMx status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

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 trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
 0: No trigger event occurred.
 1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
 Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
 Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output: This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description) and in retriggeable one pulse mode. It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT has matched the content of the TIMx_CCR1 register.

If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred.

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

At overflow or underflow (for TIM2 to TIM4) and if UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

21.4.6 TIMx event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

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

21.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

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]
							Res.								Res.
							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]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]			IC1PSC[1:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bits 16 **OC1M[3]**: Output Compare 1 mode - bit 3

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: 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

OC1CE: Output Compare 1 Clear Enable

0: OC1Ref is not affected by the ETRF input

1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

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: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=00 (the channel is configured in output).

2: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=00 (the channel is configured in output).

2: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
 0001: $f_{SAMPLING}=f_{CK_INT}$, $N=2$
 0010: $f_{SAMPLING}=f_{CK_INT}$, $N=4$
 0011: $f_{SAMPLING}=f_{CK_INT}$, $N=8$
 0100: $f_{SAMPLING}=f_{DTS}/2$, $N=6$
 0101: $f_{SAMPLING}=f_{DTS}/2$, $N=8$
 0110: $f_{SAMPLING}=f_{DTS}/4$, $N=6$
 0111: $f_{SAMPLING}=f_{DTS}/4$, $N=8$
 1000: $f_{SAMPLING}=f_{DTS}/8$, $N=6$
 1001: $f_{SAMPLING}=f_{DTS}/8$, $N=8$
 1010: $f_{SAMPLING}=f_{DTS}/16$, $N=5$
 1011: $f_{SAMPLING}=f_{DTS}/16$, $N=6$
 1100: $f_{SAMPLING}=f_{DTS}/16$, $N=8$
 1101: $f_{SAMPLING}=f_{DTS}/32$, $N=5$
 1110: $f_{SAMPLING}=f_{DTS}/32$, $N=6$
 1111: $f_{SAMPLING}=f_{DTS}/32$, $N=8$

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as $CC1E=0$ (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
 01: capture is done once every 2 events
 10: capture is done once every 4 events
 11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
 01: CC1 channel is configured as input, IC1 is mapped on TI1
 10: CC1 channel is configured as input, IC1 is mapped on TI2
 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF ($CC1E = 0$ in TIMx_CCER).

21.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

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]
							Res.								Res.
							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]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]			IC3PSC[1:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC4M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bit 16 **OC3M[3]**: Output Compare 1 mode - bit 3

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 14:12 **OC4M**: 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**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode

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**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

21.4.9 TIMx capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
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 TI1FP1/TI2FP1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*

CC1 channel configured as output:

0: OC1 active high

1: OC1 active low

CC1 channel configured as input: CC1NP/CC1P bits select TI1FP1 and TI2FP1 polarity for trigger or capture operations.

00: noninverted/rising edge

Circuit is sensitive to TIxFP1 rising edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode, encoder mode).

01: inverted/falling edge

Circuit is sensitive to TIxFP1 falling edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is inverted (trigger in gated mode, encoder mode).

10: reserved, do not use this configuration.

11: noninverted/both edges

Circuit is sensitive to both TIxFP1 rising and falling edges (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode). This configuration must not be used for encoder mode.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*

CC1 channel configured as output:

0: Off - OC1 is not active

1: On - OC1 signal is output on the corresponding output pin

CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

0: Capture disabled

1: Capture enabled

Table 120. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output Disabled (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Polarity, OCx_EN=1

Note: The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO and AFIO registers.

21.4.10 TIMx counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31] or UIFCPY	CNT[30:16] (depending on timers)														
r/w or r	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31 Value depends on IUFREMAP in TIMx_CR1.

If IUFREMAP = 0

CNT[31]: Most significant bit of counter value (on TIM2)

Reserved on other timers

If IUFREMAP = 1

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 (on TIM2)

Bits 15:0 **CNT[15:0]:** Least significant part of counter value

21.4.11 TIMx prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **PSC[15:0]:** Prescaler value

The counter clock frequency CK_CNT is equal to $f_{CK_PSC} / (PSC[15:0] + 1)$.

PSC contains the value to be loaded in the active prescaler register at each update event.

21.4.12 TIMx auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x00000000

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

Bits 31:16 **ARR[31:16]:** High auto-reload value (on TIM2)

Bits 15:0 **ARR[15:0]:** Low Auto-reload Prescaler 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 548](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

21.4.13 TIMx capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16] (depending on timers)															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR1[31:16]**: High Capture/Compare 1 value (on TIM2 and TIM5)

Bits 15:0 **CCR1[15:0]**: Low Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

21.4.14 TIMx capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16] (depending on timers)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR2[31:16]**: High Capture/Compare 2 value (on TIM2 and TIM5)

Bits 15:0 **CCR2[15:0]**: Low Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_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).

21.4.15 TIMx capture/compare register 3 (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16] (depending on timers)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (on TIM2 and TIM5)

Bits 15:0 **CCR3[15:0]**: Low Capture/Compare value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_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).

21.4.16 TIMx capture/compare register 4 (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16] (depending on timers)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR4[31:16]**: High Capture/Compare 4 value (on TIM2)

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).

21.4.17 TIMx DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

00000: 1 transfer,
 00001: 2 transfers,
 00010: 3 transfers,
 ...
 10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1
 00001: TIMx_CR2
 00010: TIMx_SMCR
 ...

Example: Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

21.4.18 TIMx DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

$$(\text{TIMx_CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4$$

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

21.4.19 TIMx register map

TIMx registers are mapped as described in the table below:

Table 121. TIM2/TIM3/TIM4 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UIFREMAP	Res	CKD [1:0]	0	0	ARPE	CMS [1:0]	DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	TIMx_CR2	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	
	Reset value																									0	0	0	0	0			
0x08	TIMx_SMCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SMS[3]	ETP	ECE	ETPS [1:0]	ETF[3:0]			MSM	TS[2:0]			Res	SMS[2:0]				
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	TIMx_DIER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0		0		0	0	0	0	0
0x10	TIMx_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC4OF	CC3OF	CC2OF	CC1OF	Res	Res	TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
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	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]			OC2PE	OC2FE	CC2S [1:0]			OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC2F[3:0]			IC2 PSC [1:0]	CC2S [1:0]			IC1F[3:0]			IC1 PSC [1:0]	CC1S [1:0]				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIMx_CCMR2 Output Compare mode	Res	Res	Res	Res	Res	Res	Res	OC4M[3]	Res	Res	Res	Res	Res	Res	Res	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]			OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC4F[3:0]			IC4 PSC [1:0]	CC4S [1:0]			IC3F[3:0]			IC3 PSC [1:0]	CC3S [1:0]				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIMx_CCER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC4NP	Res	CC4P	CC4E	CC3NP	Res	CC3P	CC3E	CC2NP	Res	CC2P	CC2E	CC1NP	Res	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0		0	0	0

[illegible]

22 General-purpose timers (TIM15/16/17)

22.1 TIM15/16/17 introduction

The TIM15/16/17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/16/17 timers are completely independent, and do not share any resources. They can be synchronized together as described in [Timer synchronization on page 585](#).

22.2 TIM15 main features

TIM15 includes the following features:

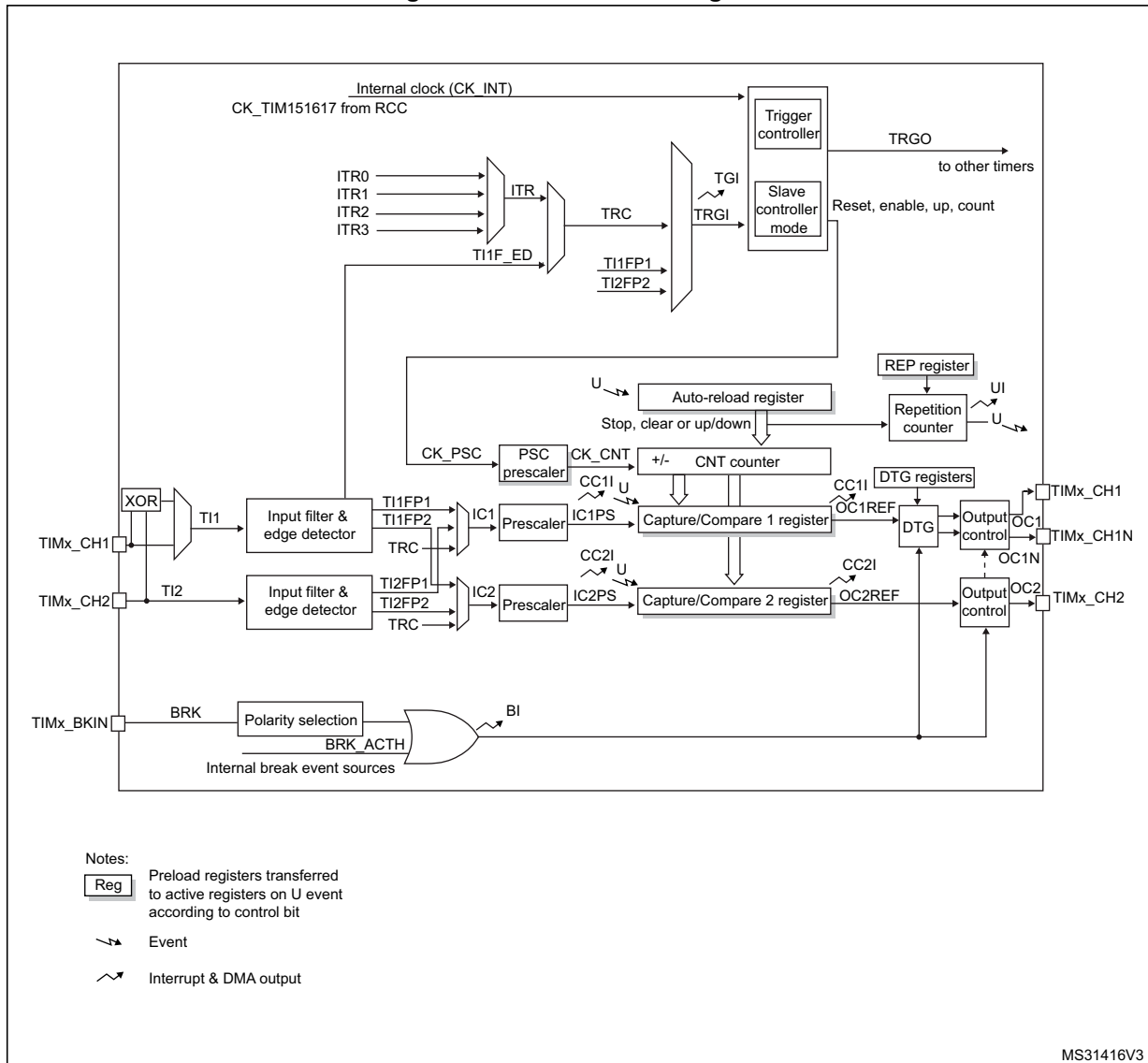
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
 - Input capture
 - Output compare
 - PWM generation (edge mode)
 - One-pulse mode output
- Complementary 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)

22.3 TIM16 and TIM17 main features

The TIM16 and TIM17 timers include the following features:

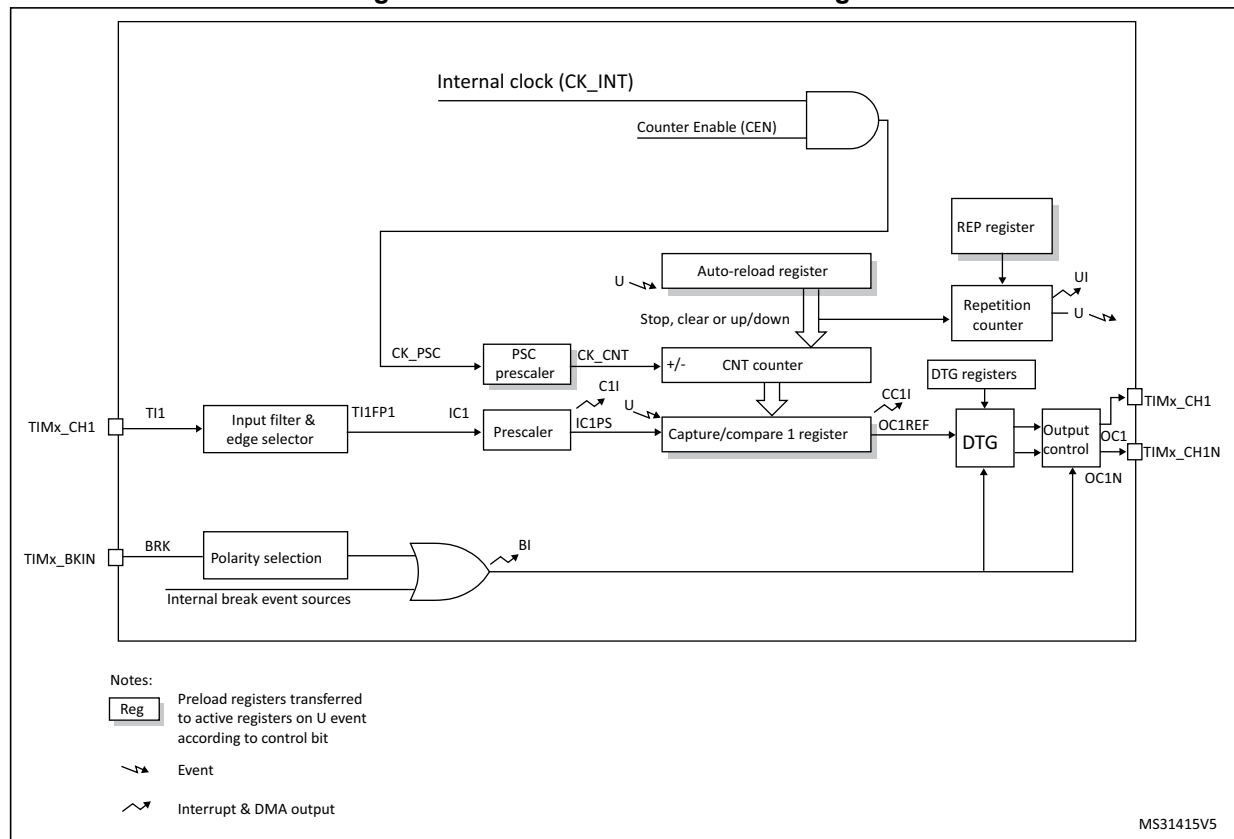
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
 - Input capture
 - Output compare
 - PWM generation (edge-aligned mode)
 - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
 - Update: counter overflow
 - Trigger event (counter start, stop, initialization or count by internal/external trigger)
 - Input capture
 - Output compare
 - Break input

Figure 237. 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 10.2.7: Clock security system \(CSS\)](#)
 - A PVD output
 - SRAM parity error signal
 - Cortex-M4® F LOCKUP (Hardfault) output
 - COMP output

Figure 238. TIM16 and TIM17 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 10.2.7: Clock security system \(CSS\)](#)
 - A PVD output
 - SRAM parity error signal
 - Cortex-M4[®]F LOCKUP (Hardfault) output
 - COMP output

22.4 TIM15/16/17 functional description

22.4.1 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 239 and *Figure 240* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 239. Counter timing diagram with prescaler division change from 1 to 2

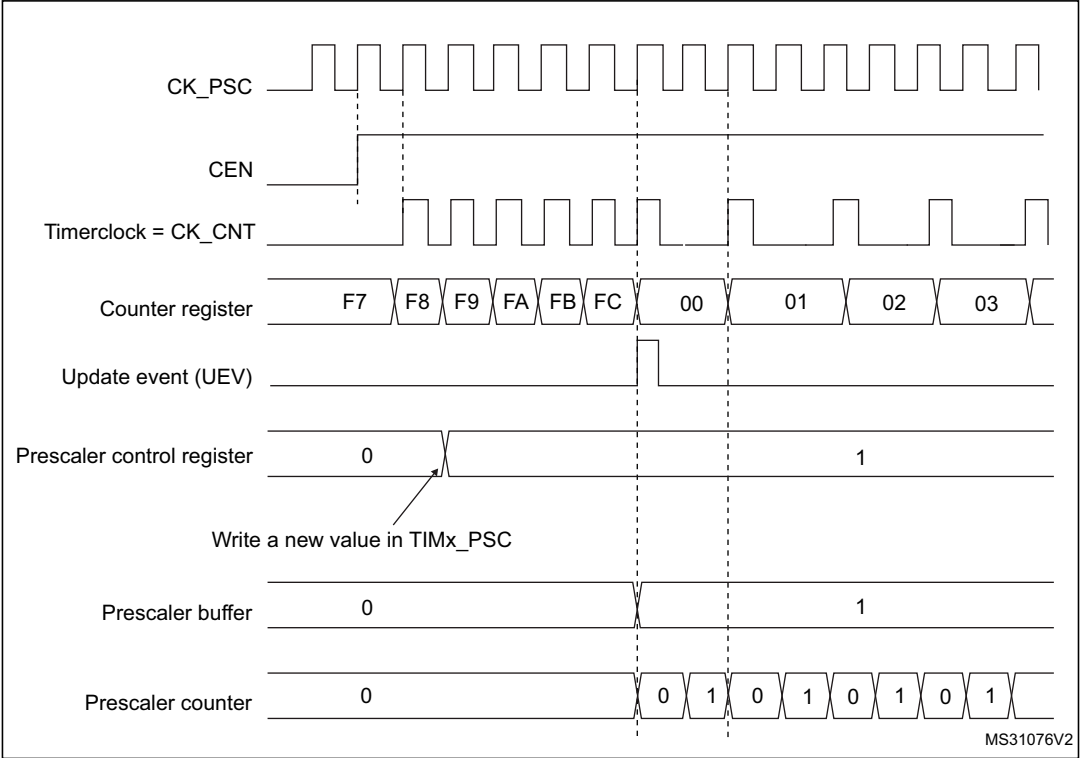
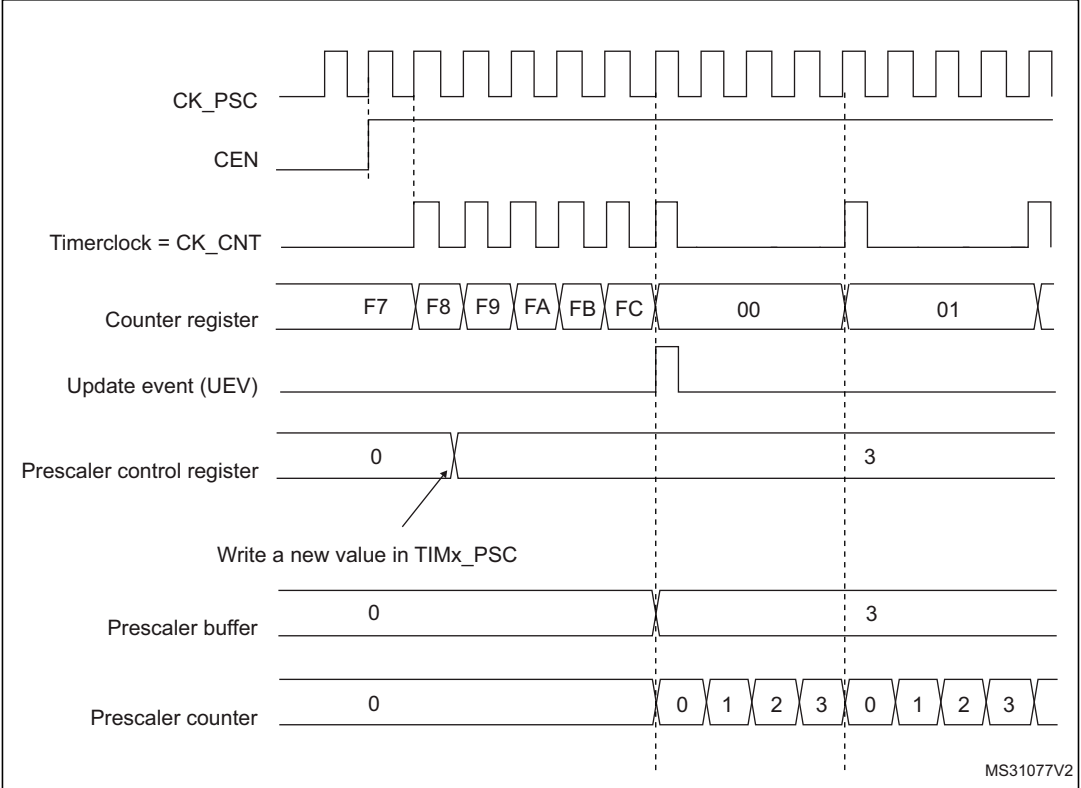


Figure 240. Counter timing diagram with prescaler division change from 1 to 4



22.4.2 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 241. Counter timing diagram, internal clock divided by 1

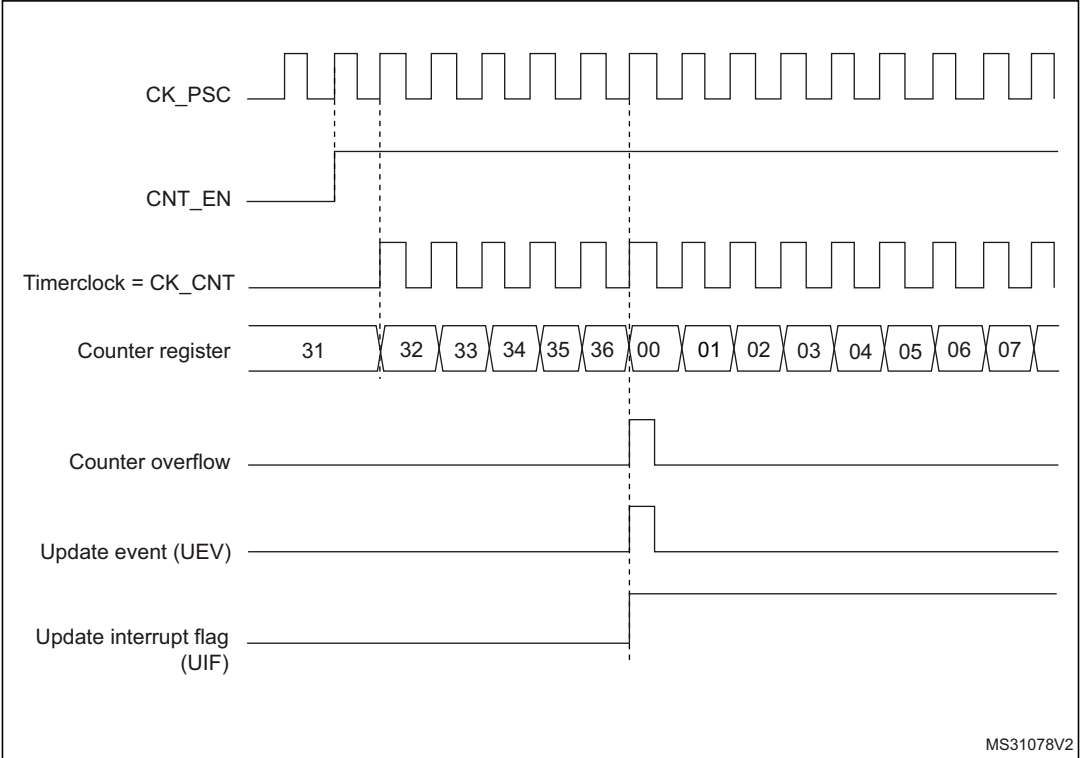


Figure 242. Counter timing diagram, internal clock divided by 2

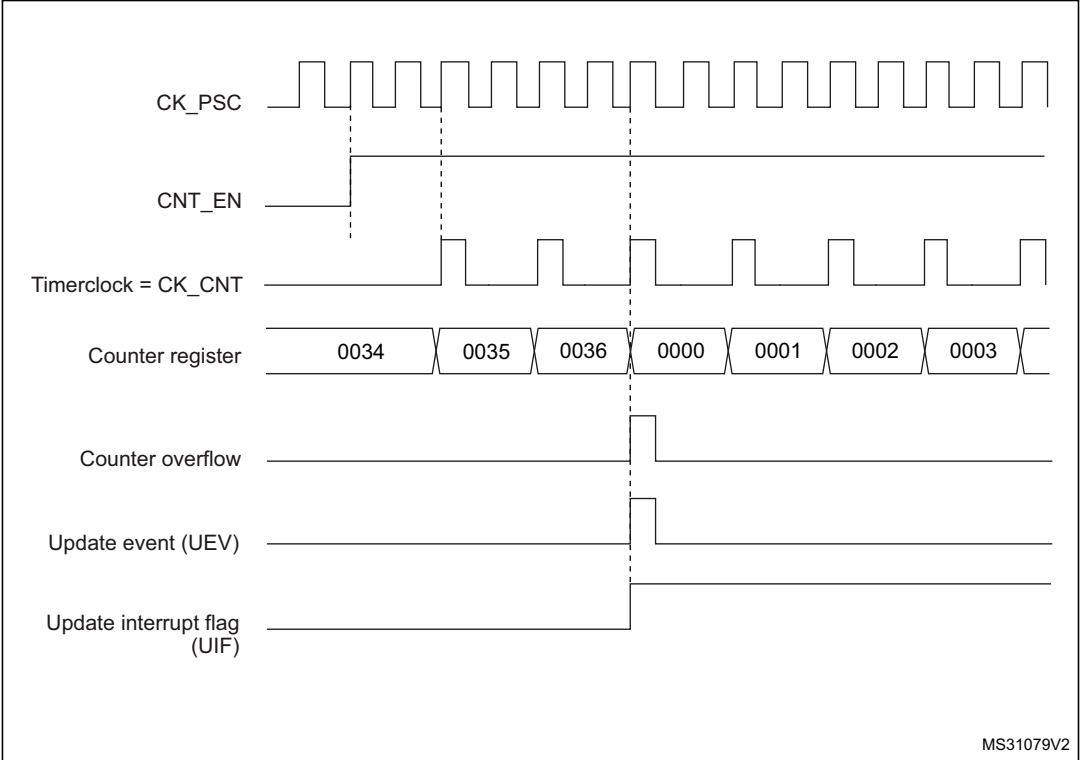


Figure 243. Counter timing diagram, internal clock divided by 4

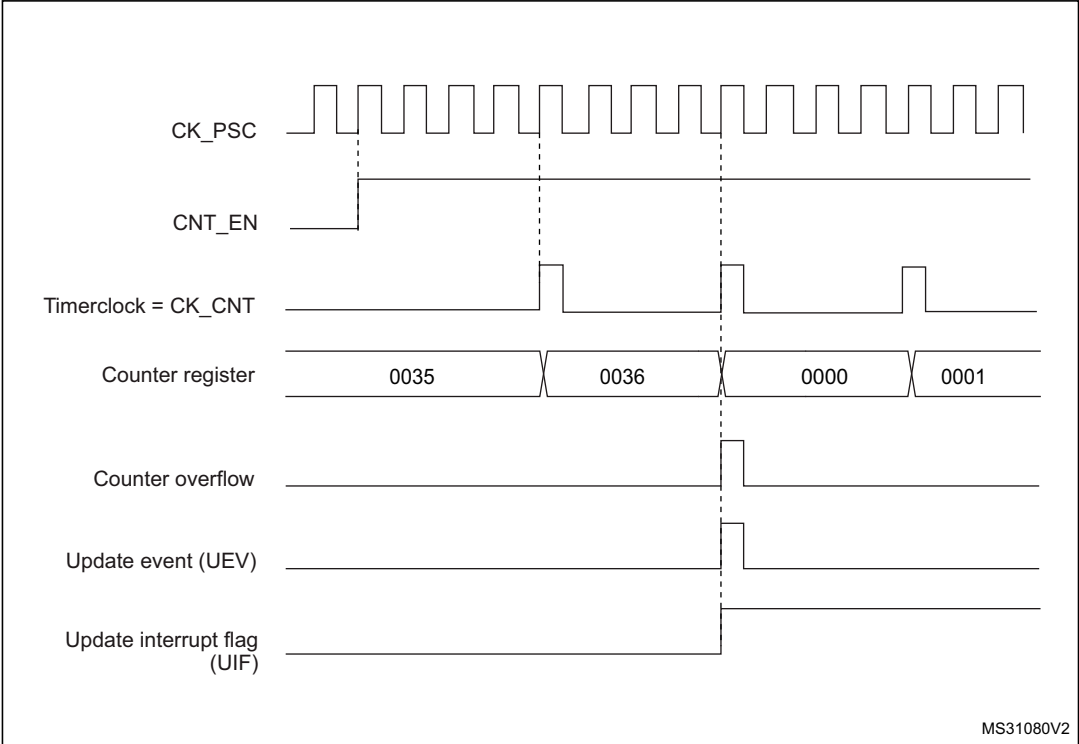


Figure 244. Counter timing diagram, internal clock divided by N

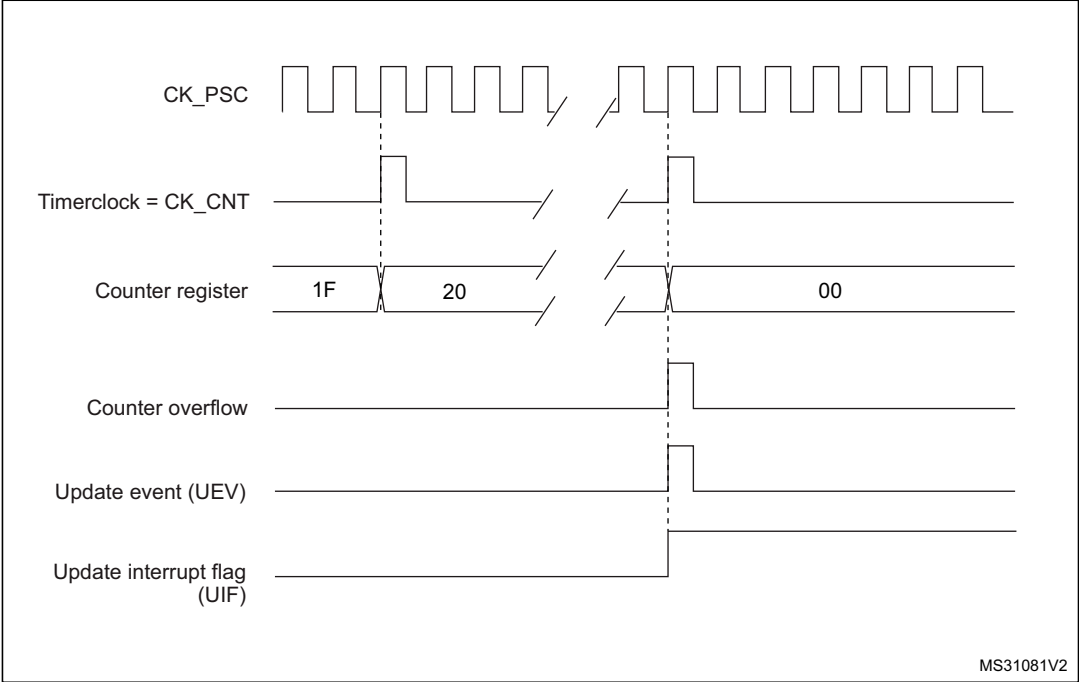


Figure 245. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

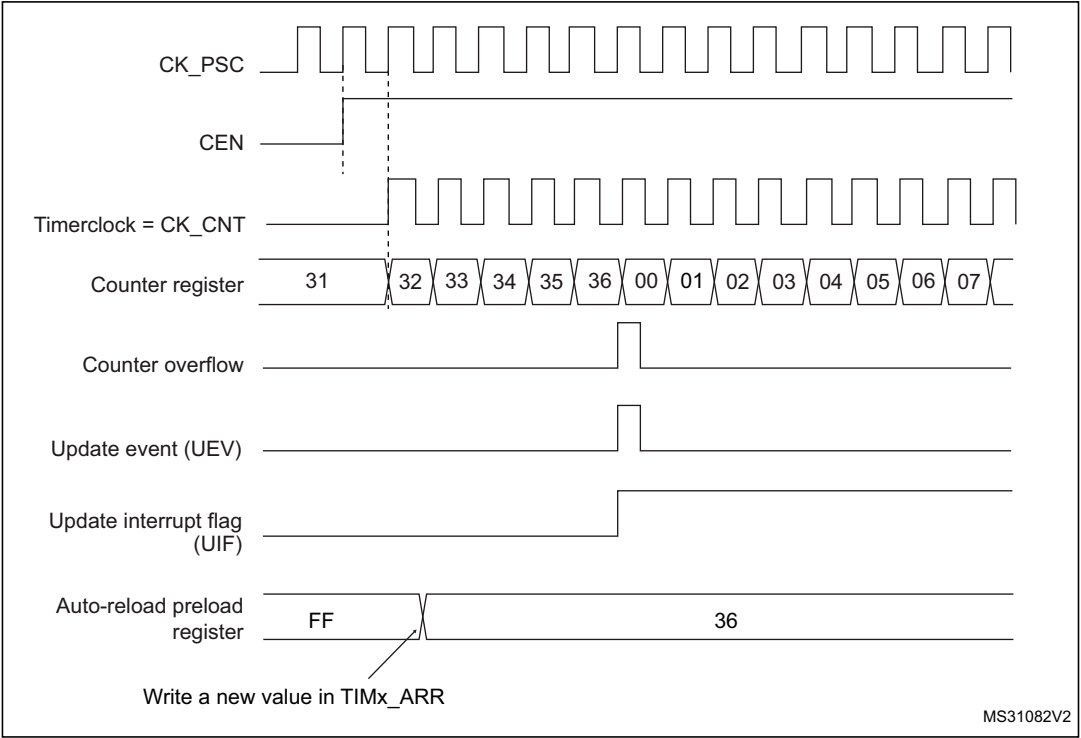
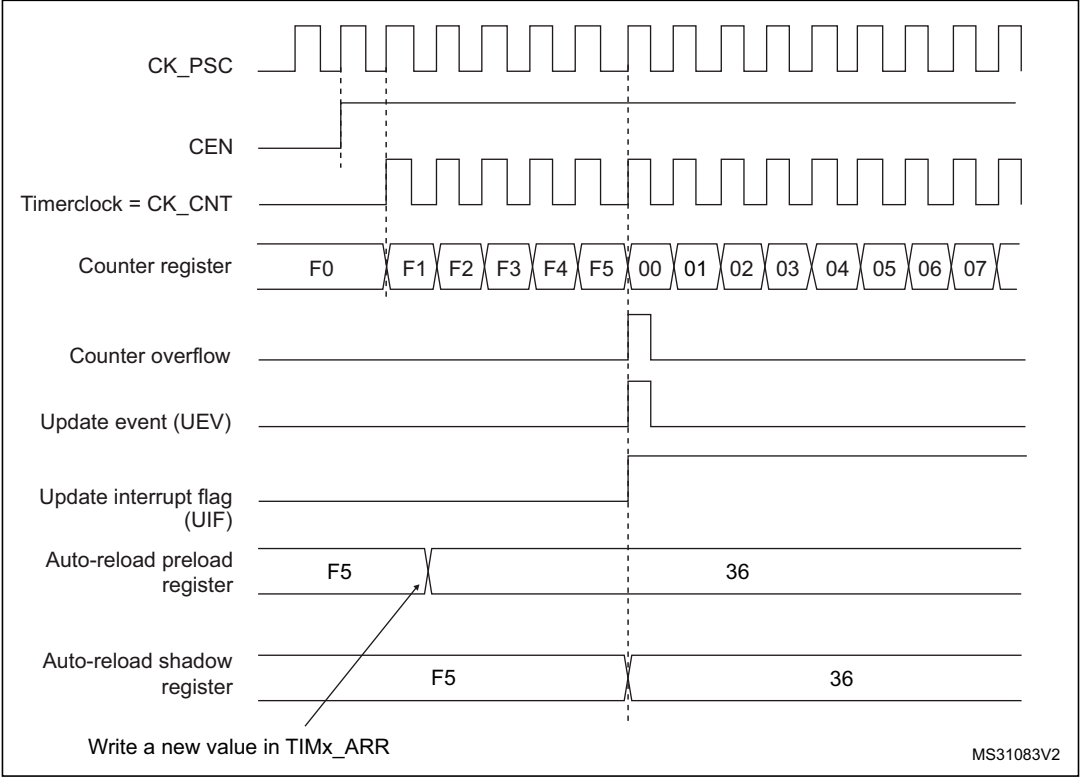


Figure 246. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



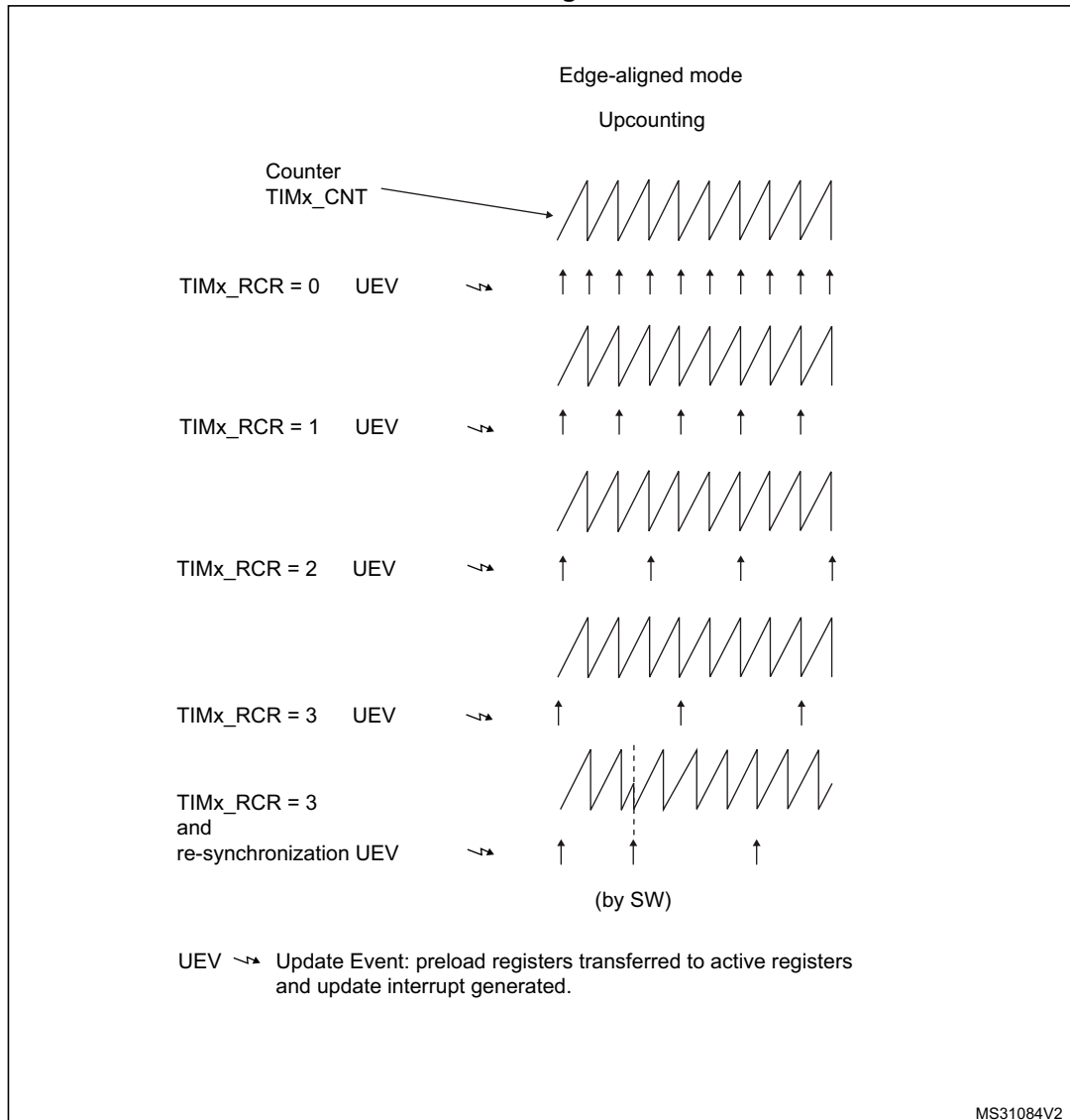
22.4.3 Repetition counter

Section 22.4.1: 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 247*). 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 247. Update rate examples depending on mode and TIMx_RCR register settings

22.4.4 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin
- Internal trigger inputs (ITRx) (only for TIM15): using one timer as the prescaler for another timer, for example, you can configure TIM1 to act as a prescaler for TIM15. Refer to [Using one timer as prescaler for another on page 586](#) for more details.

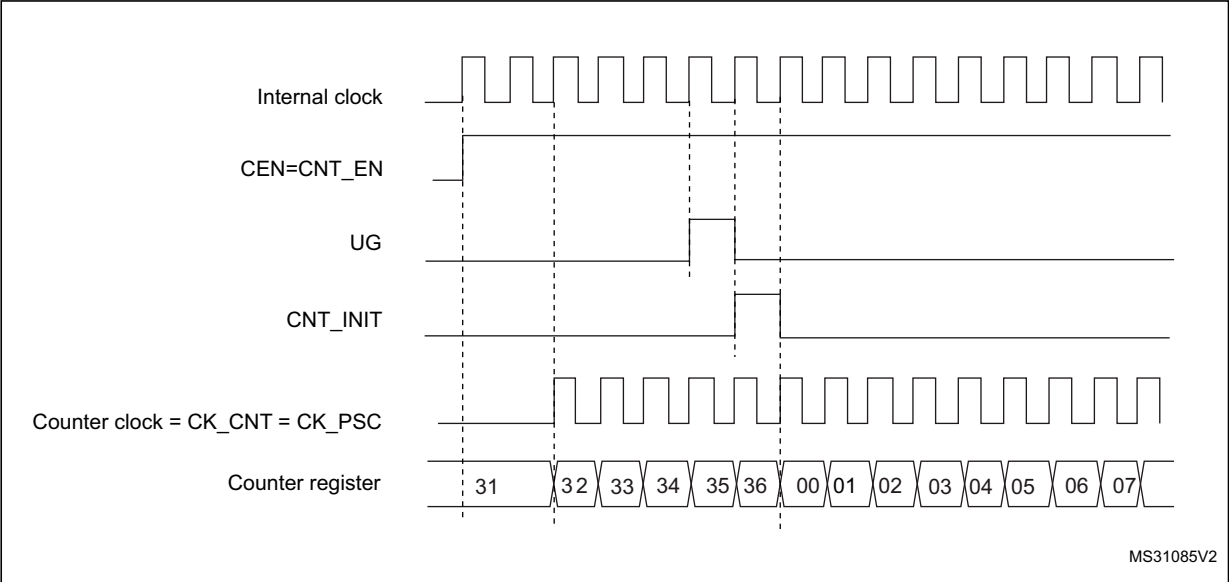
Internal clock source (CK_INT)

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 248 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

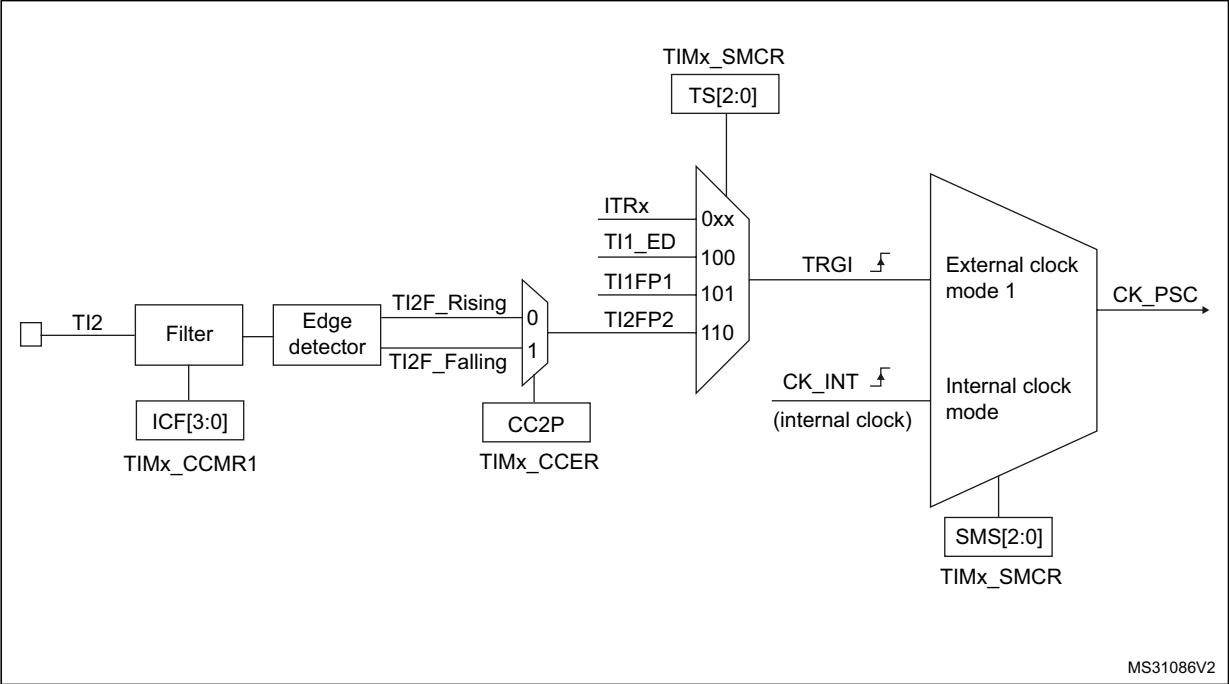
Figure 248. Control circuit in normal mode, internal clock divided by 1



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 249. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

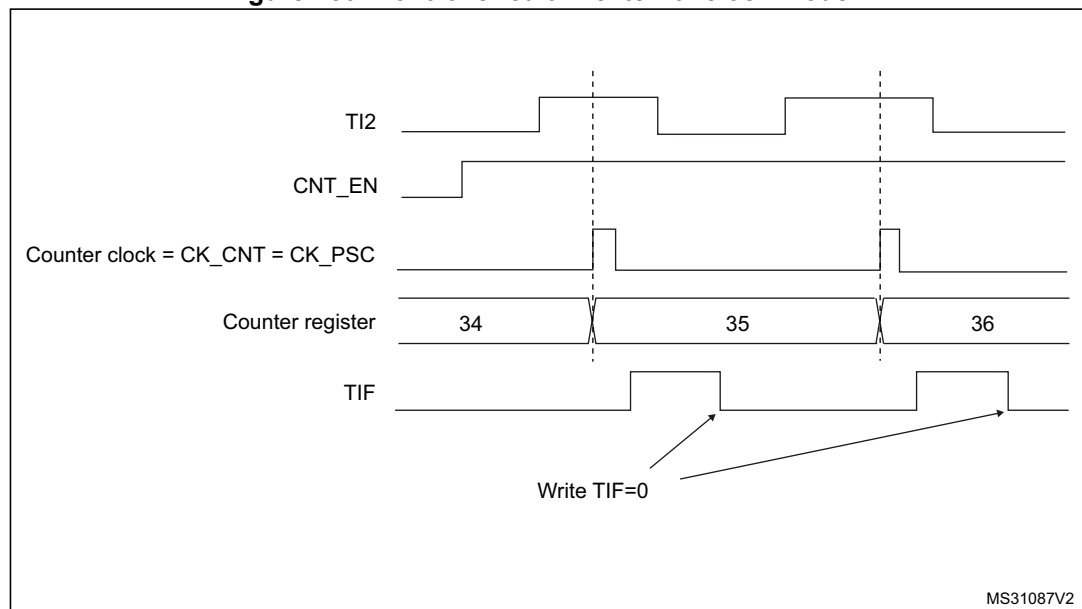
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the trigger input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: *The capture prescaler is not used for triggering, so you don't need to configure it.*

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 250. Control circuit in external clock mode 1



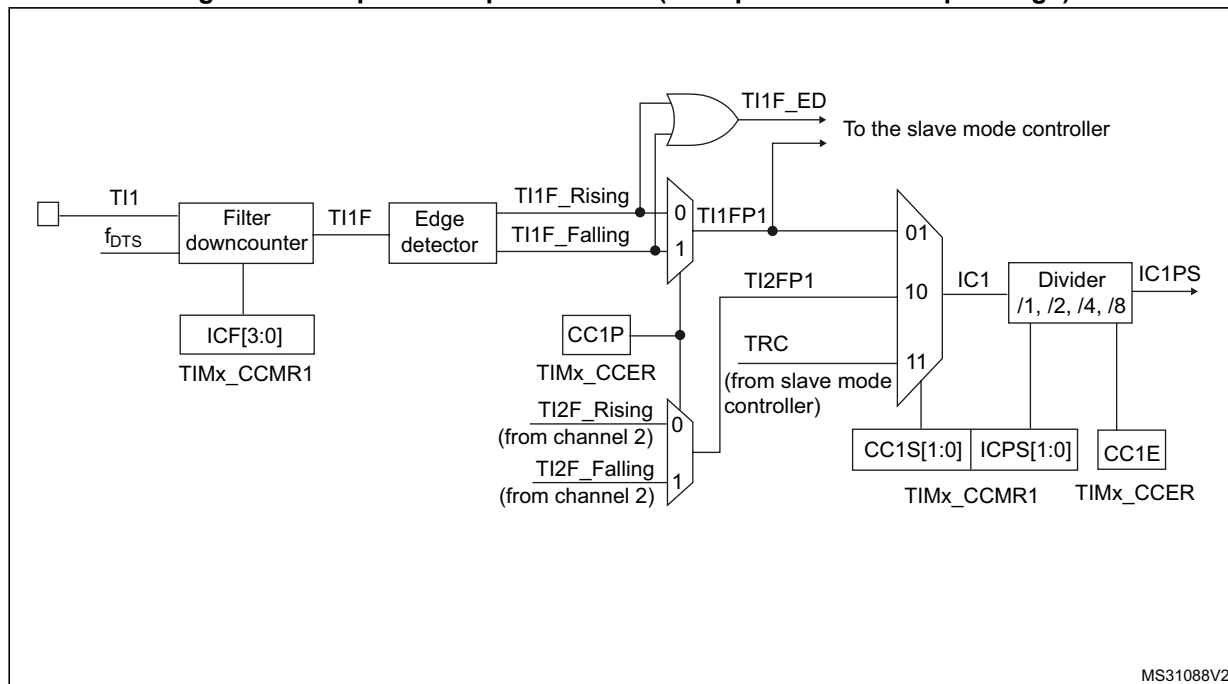
22.4.5 Capture/compare channels

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 251 to Figure 254 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 251. 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 252. Capture/compare channel 1 main circuit

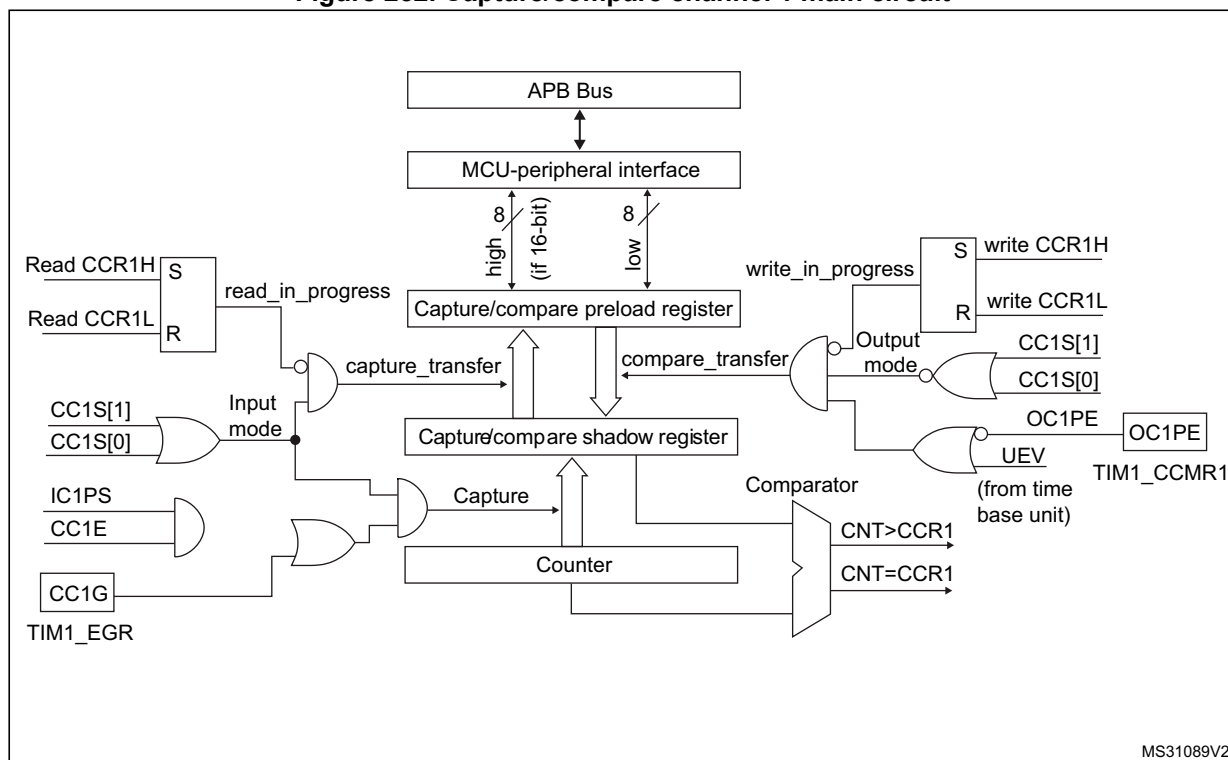
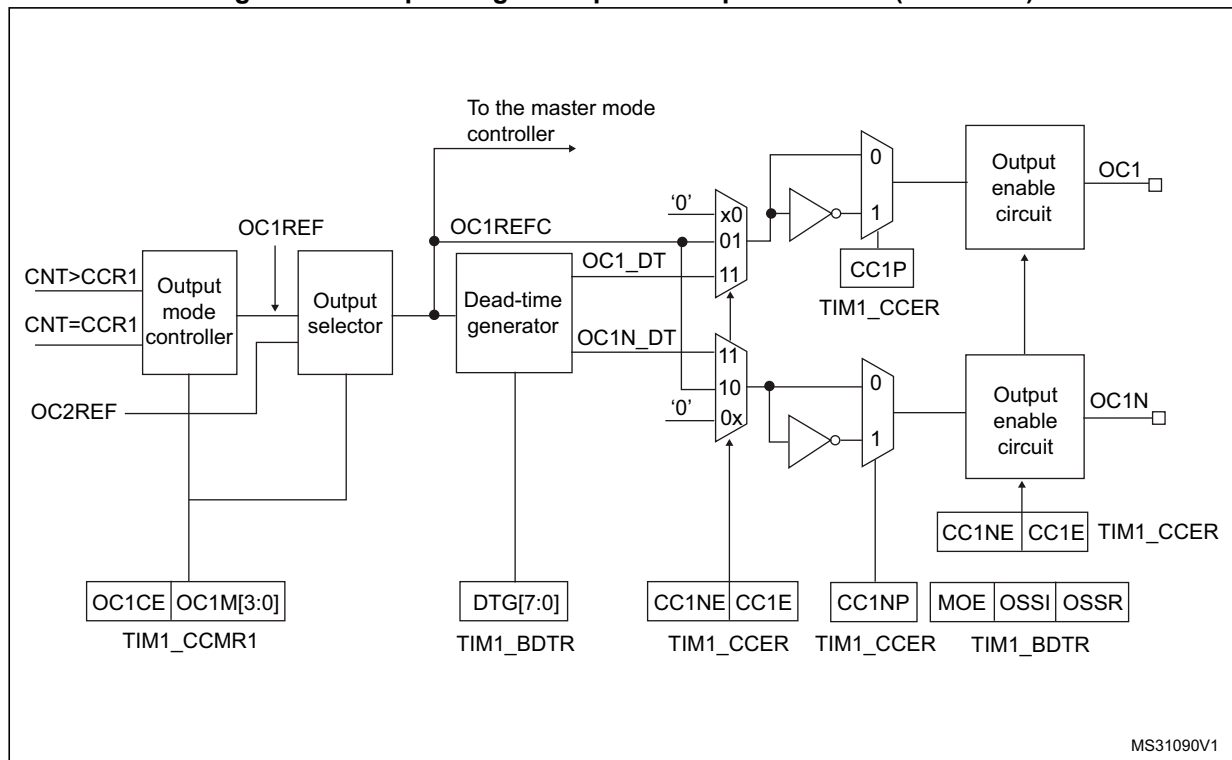
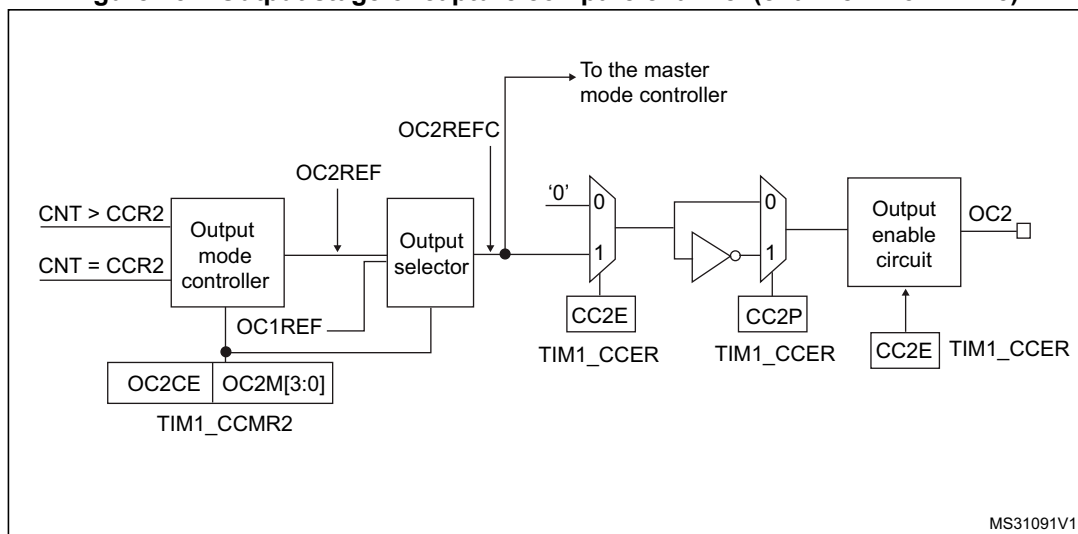


Figure 253. Output stage of capture/compare channel (channel 1)



MS31090V1

Figure 254. Output stage of capture/compare channel (channel 2 for TIM15)



MS31091V1

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.

22.4.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
2. Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
3. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

22.4.7 PWM input mode (only for TIM15)

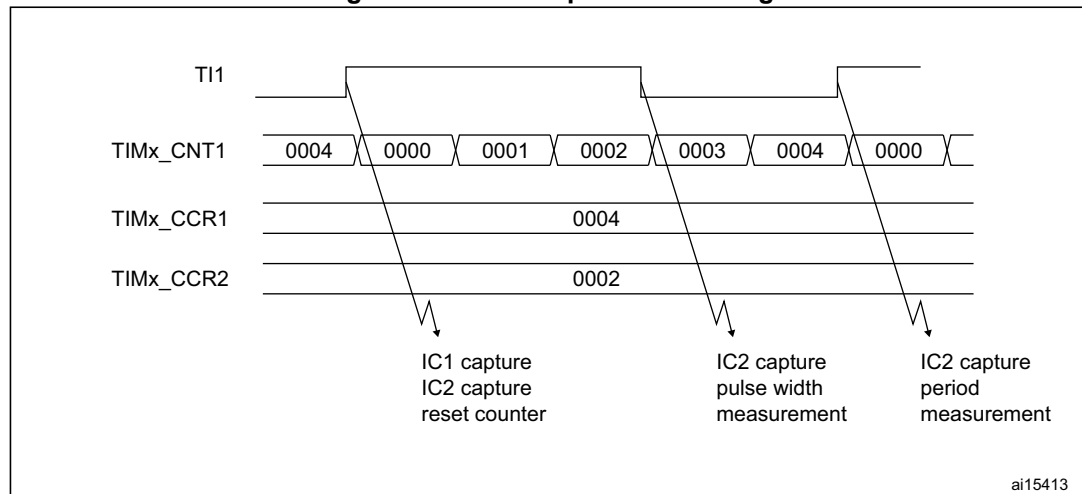
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to '1' (active on falling edge).
5. Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 255. 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.

22.4.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

22.4.9 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).

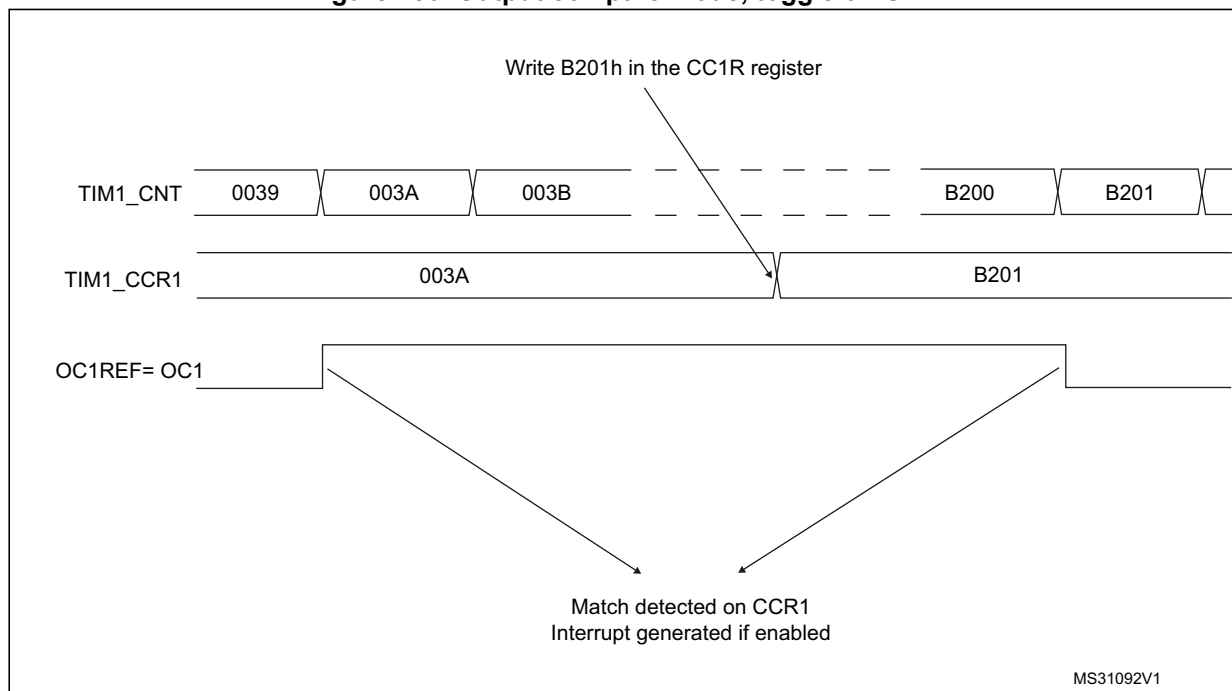
The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

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 256. Output compare mode, toggle on OC1**22.4.10 PWM mode**

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

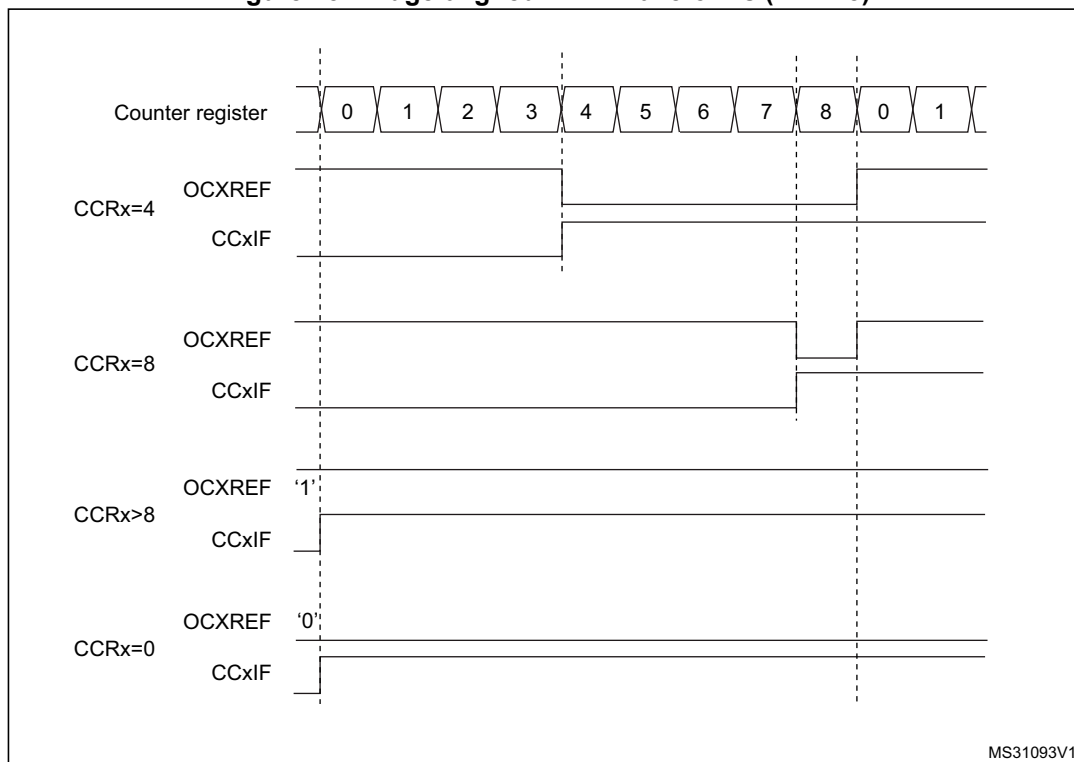
OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether $TIMx_CCRx \leq TIMx_CNT$ or $TIMx_CNT \leq TIMx_CCRx$ (depending on the direction of the counter).

The TIM15/16/17 are capable of upcounting only. Refer to [Upcounting mode on page 622](#).

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 257](#) shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 257. Edge-aligned PWM waveforms (ARR=8)



22.4.11 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.

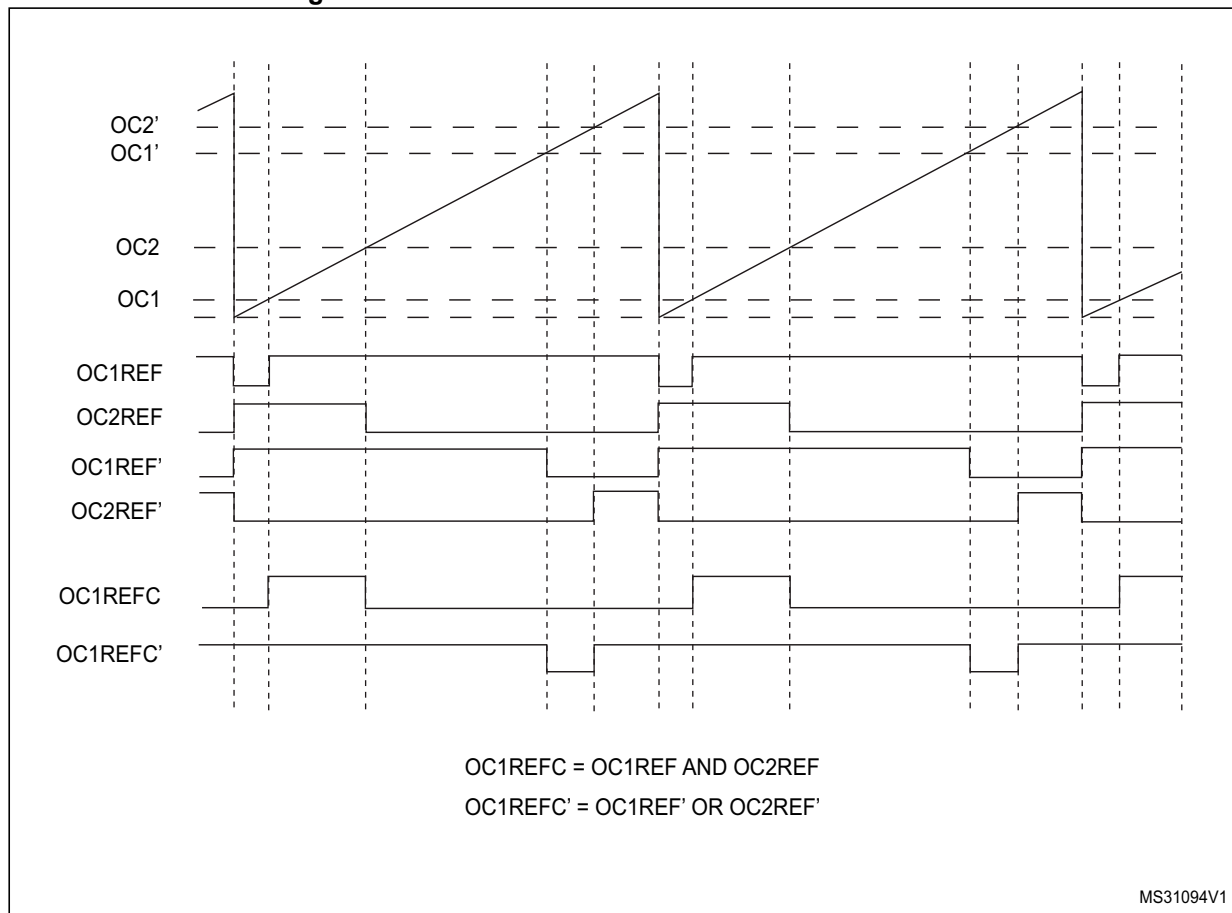
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

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 258 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 258. Combined PWM mode on channel 1 and 2



22.4.12 Complementary outputs and dead-time insertion

The TIM15/16/17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and you have to adjust it depending on the devices you have connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

You can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to [Table 123: Output control bits for complementary OCx and OCxN channels with break feature on page 665](#) 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 259. Complementary output with dead-time insertion.

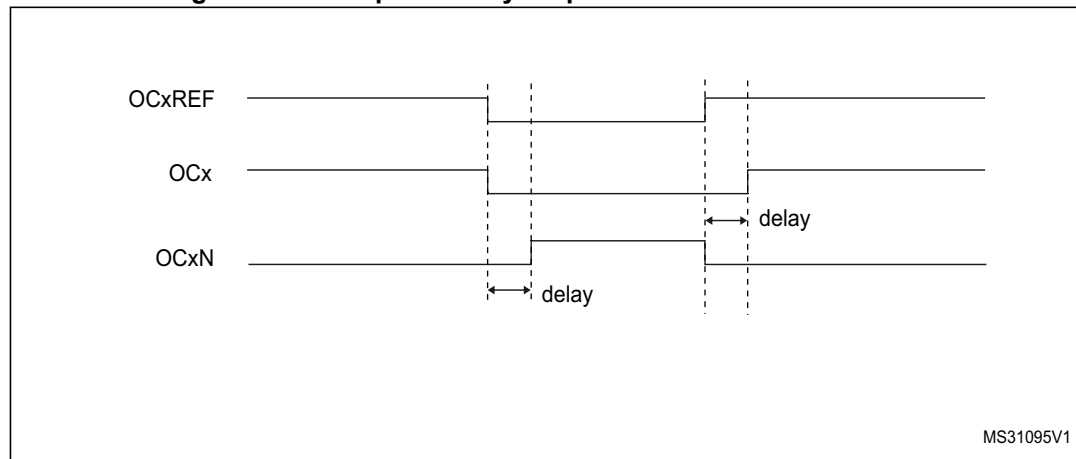
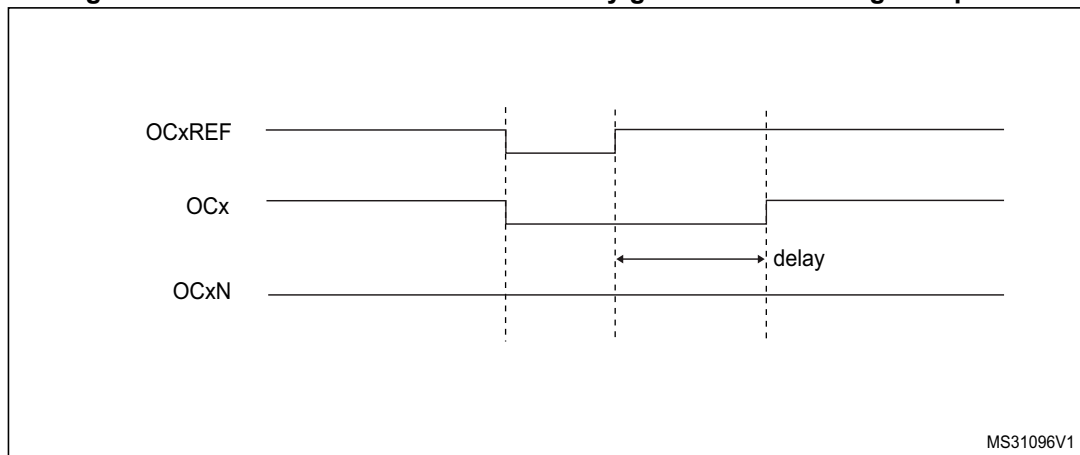
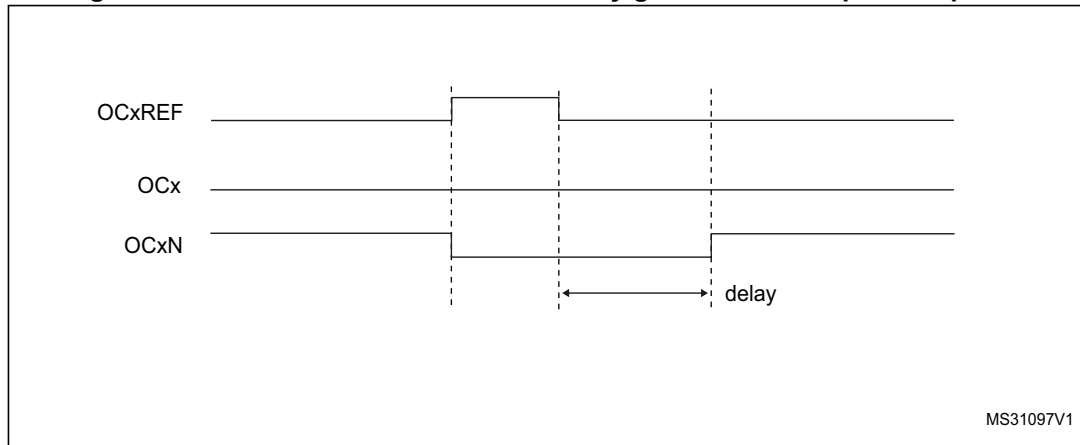


Figure 260. Dead-time waveforms with delay greater than the negative pulse.**Figure 261. Dead-time waveforms with delay greater than the positive pulse.**

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 22.5.15: TIM15 break and dead-time register \(TIM15_BDTR\) on page 668](#) for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows you to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

22.4.13 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM15/16/17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, OISx and OISxN bits in the TIMx_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to [Table 123: Output control bits for complementary OCx and OCxN channels with break feature on page 665](#) for more details.

The break source can be:

- An external source connected to BKIN pin (connected internally to BRK)
- An internal source (connected internally to BRK_ACTH):
 - A clock failure event generated by CSS. For further information on the CSS, refer to [Section 10.2.7: Clock security system \(CSS\)](#)
 - An output from a comparator
 - A PVD output
 - SRAM parity error signal
 - Cortex[®]-M4 LOCKUP (Hardfault) output

When exiting from reset, the break circuit is disabled and the MOE bit is low. You can enable the break function by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if you write MOE to 1 whereas it was low, you must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read the synchronous signal.

The break is generated by the BRK inputs which has:

- Programmable polarity (BKP bit in the TIMx_BDTR register)
- Programmable enable bit (BKE bit in the TIMx_BDTR register)
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

It is also possible to generate break events by software using BG bit in TIMx_EGR register.

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 AFIO controller (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 AFIO controller) else the enable output remains high.
- When complementary outputs are used:
 - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
 - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).
 - If OSS1=0 then the timer releases the enable outputs (taken over by the AFIO controller 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. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until you write it to '1' again. In this case, it can be used for security and you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

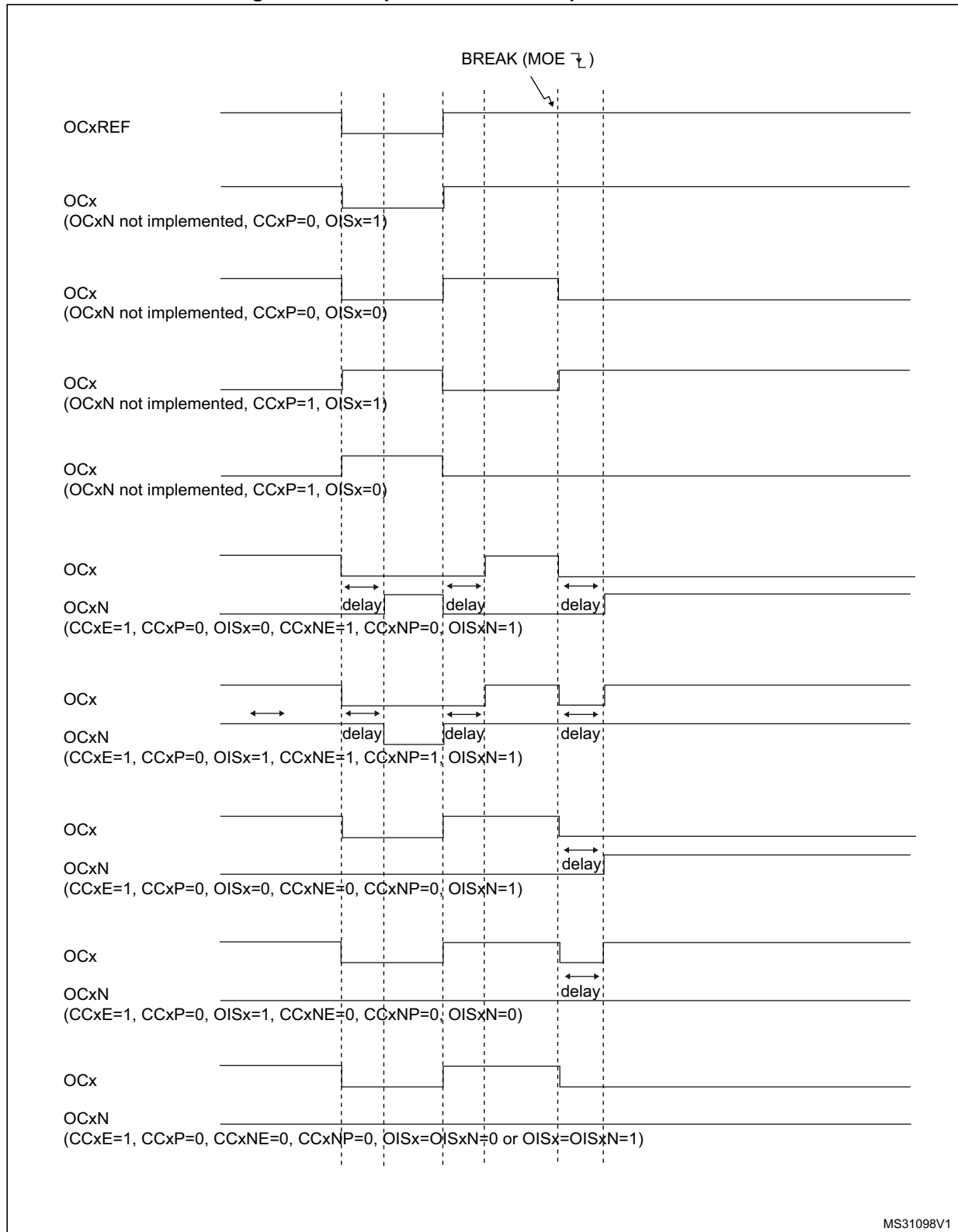
Note: *The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.*

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows you to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). You can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to [Section 22.5.15: TIM15 break and dead-time register \(TIM15_BDTR\) on page 668](#). 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



22.4.14 One-pulse mode

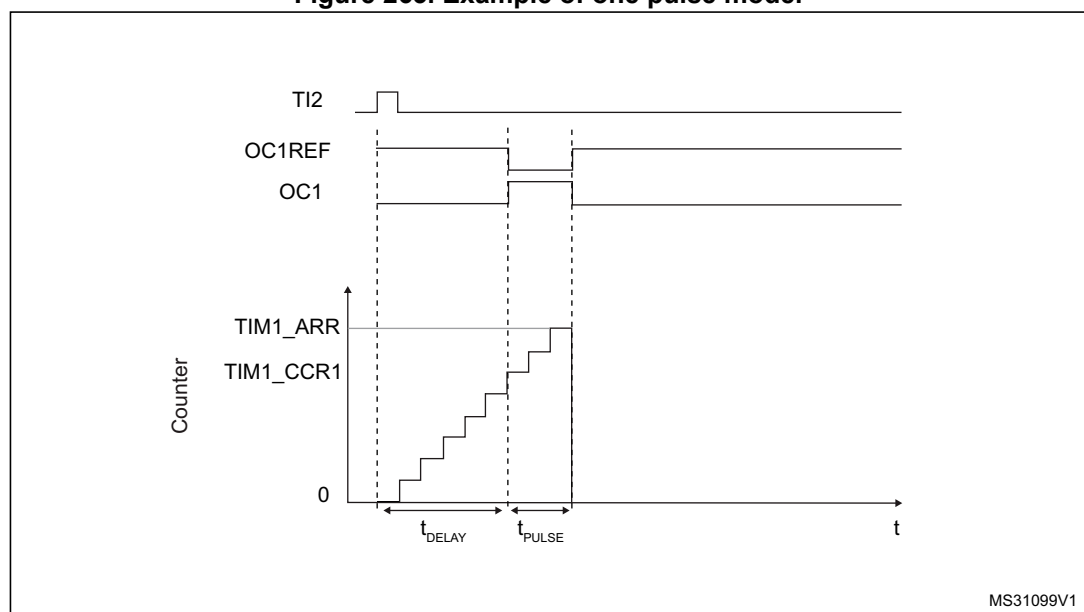
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $CNT < CCRx \leq ARR$ (in particular, $0 < CCRx$)

Figure 263. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
2. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
4. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

You only want 1 pulse, so you write '1' in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

22.4.15 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows 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).

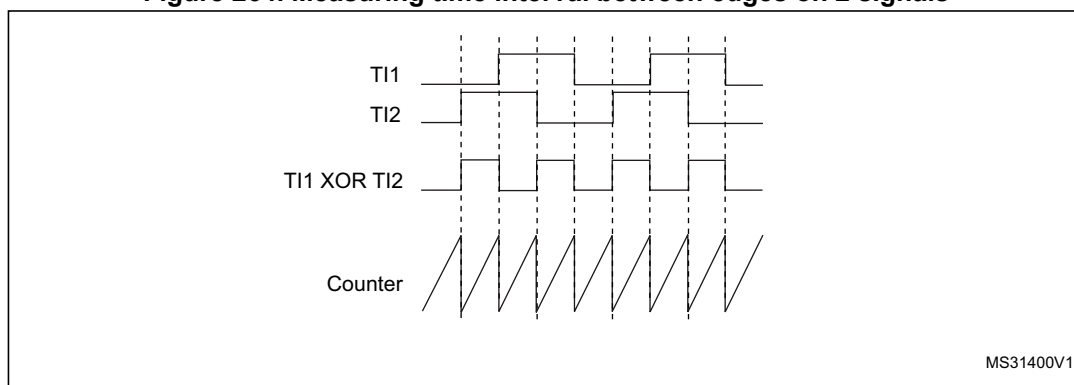
There is no latency between the assertions of the UIF and UIFCPY flags.

22.4.16 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 264](#).

Figure 264. Measuring time interval between edges on 2 signals



22.4.17 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.

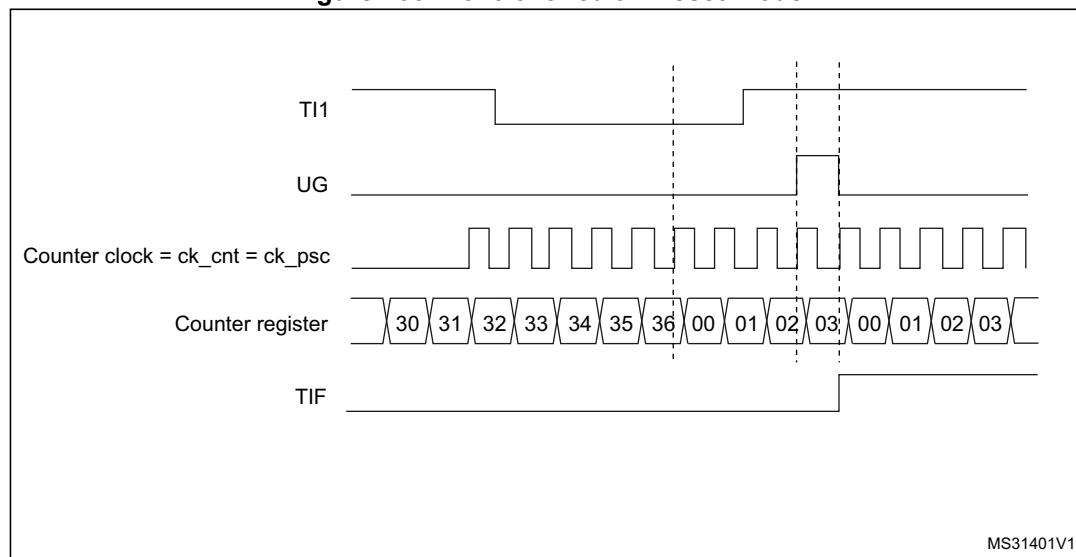
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P='0' and CC1NP='0' in 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=101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 265. Control circuit in reset mode



MS31401V1

Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

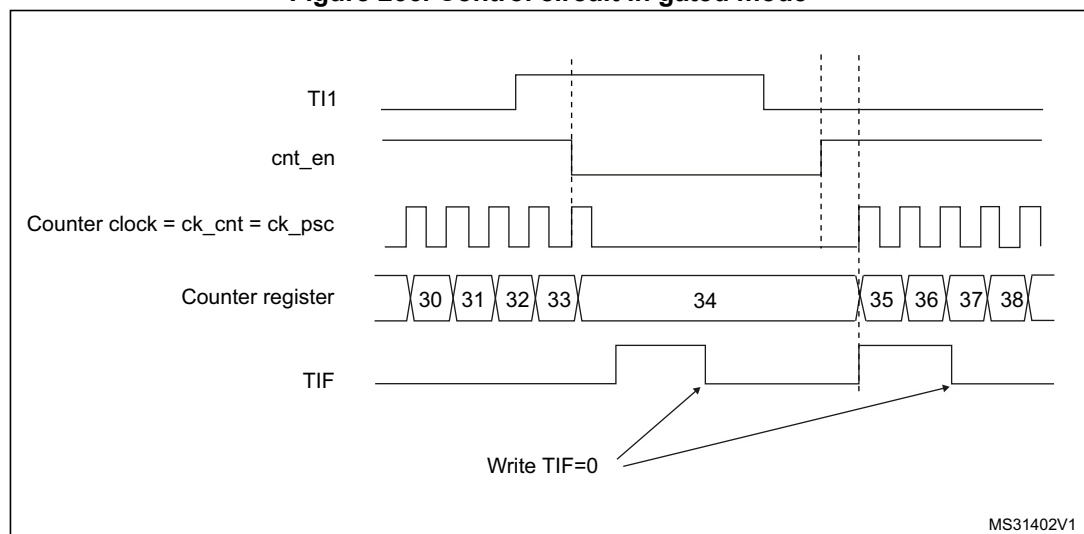
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP = '0' in 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=101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 266. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

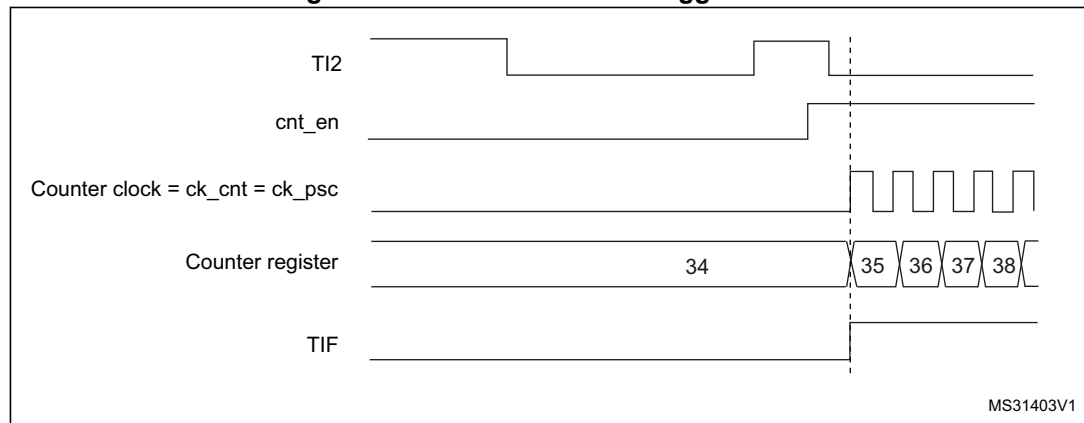
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P='1' 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=110 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 267. Control circuit in trigger mode



22.4.18 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.

22.4.19 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.

22.4.20 Debug mode

When the microcontroller enters debug mode (Cortex-M4[®]F 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 33.14.2: Debug support for timers, watchdog, bxCAN and I2C](#).

22.5 TIM15 registers

Refer to [Section 2.1](#) for a list of abbreviations used in register descriptions.

22.5.1 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
				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 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 (Tlx)

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.

22.5.2 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

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

Bit 10 **OIS2**: Output idle state 2 (OC2 output)

0: OC2=0 when MOE=0

1: OC2=1 when MOE=0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIMx_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 7 **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[1:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

- 000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
- 001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
- 010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
- 011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
- 100: **Compare** - OC1REF signal is used as trigger output (TRGO).
- 101: **Compare** - OC2REF signal is used as trigger output (TRGO).

Bit 3 **CCDS**: Capture/compare DMA selection

- 0: CCx DMA request sent when CCx event occurs
- 1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

- 0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
- 1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

- 0: CCxE, CCxNE and OCxM bits are not preloaded
- 1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when 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.

22.5.3 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.	Res.	Res.	Res.	Res.	Res.	SMS[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.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

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

Bits 16 **SMS[3]**: Slave mode selection - bit 3

Refer to SMS description - bits 2:0

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

Bit 7 **MSM**: Master/slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]**: Trigger selection

This bit field selects the trigger input to be used to synchronize the counter.

- 000: Internal Trigger 0 (ITR0)
- 001: Internal Trigger 1 (ITR1)
- 010: Internal Trigger 2 (ITR2)
- 011: Internal Trigger 3 (ITR3)
- 100: TI1 Edge Detector (TI1F_ED)
- 101: Filtered Timer Input 1 (TI1FP1)
- 110: Filtered Timer Input 2 (TI2FP2)

See [Table 122: TIMx Internal trigger connection on page 654](#) for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMS**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

- 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='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Table 122. TIMx Internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM15	TIM2	TIM3	TIM16 OC1	TIM17 OC1

22.5.4 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.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	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

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable

0: CC2 DMA request disabled

1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable

0: CC1 DMA request disabled

1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled

1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable

0: Break interrupt disabled

1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable

0: Trigger interrupt disabled

1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable

0: COM interrupt disabled

1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable

0: CC2 interrupt disabled

1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled

1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

1: Update interrupt enabled

22.5.5 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 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 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

If channel CC1 is configured as output: This flag is set by hardware when the counter matches the compare value. It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.

If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow 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 22.5.3: TIM15 slave mode control register \(TIM15_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx_CR1 register.

22.5.6 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).

22.5.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

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]
							Res.								Res.
							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]	
IC2F[3:0]			IC2PSC[1:0]		IC1F[3:0]			IC1PSC[1:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, always read as 0

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, always read as 0

Bit 16 **OC1M[3]**: Output Compare 1 mode - bit 3
refer to OC1M description on bits 6:4

Bit 15 Reserved, always read as 0

Bits 14:12 **OC2M[2:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 Reserved, always read as 0

Bits 6:4 **OC1M**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

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: Reserved,

1001: Reserved,

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: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

2: In PWM mode, 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.

3: 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.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
 0001: $f_{SAMPLING}=f_{CK_INT}$, N=2
 0010: $f_{SAMPLING}=f_{CK_INT}$, N=4
 0011: $f_{SAMPLING}=f_{CK_INT}$, N=8
 0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
 0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
 0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
 0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
 1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
 1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
 1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
 1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
 1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
 1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
 1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
 1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
 01: capture is done once every 2 events
 10: capture is done once every 4 events
 11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
 01: CC1 channel is configured as input, IC1 is mapped on TI1
 10: CC1 channel is configured as input, IC1 is mapped on TI2
 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

22.5.8 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw	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 **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: 1. 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).
2. 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

CC1 channel configured as output:

0: OC1 active high

1: OC1 active low

CC1 channel configured as input: The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

00: 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).

01: 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).

10: reserved, do not use this configuration.

11: 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).

Note: 1. This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

2. 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

CC1 channel configured as output:

0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

0: Capture disabled

1: Capture enabled

Table 123. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states ⁽¹⁾	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	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	OCx=CCxP, OCxN=CCxNP	
			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 AFIO registers.

22.5.9 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]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

22.5.10 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to $f_{CK_PSC} / (PSC[15:0] + 1)$.

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

22.5.11 TIM15 auto-reload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 22.4.1: Time-base unit on page 620](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

22.5.12 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

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

22.5.13 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
r/w	r/w	r/w	r/w	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).

22.5.14 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

22.5.15 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	Res	Res	Res	Res	Res	Res	Res	Res	Res	BKF[3:0]			
												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 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 31:20 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 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 22.5.8: TIM15 capture/compare enable register \(TIM15_CCER\) on page 662](#)).

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: 1: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

2: 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 22.5.8: TIM15 capture/compare enable register \(TIM15_CCER\) on page 662](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO 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 on channels configured as outputs.

See OC/OCN enable description for more details ([Section 22.5.8: TIM15 capture/compare enable register \(TIM15_CCER\) on page 662](#)).

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 $T_{dtg}=2xt_{DTS}$

DTG[7:5]=110 => DT=(32+DTG[4:0])x t_{dtg} with $T_{dtg}=8xt_{DTS}$

DTG[7:5]=111 => DT=(32+DTG[4:0])x t_{dtg} with $T_{dtg}=16xt_{DTS}$

Example if $T_{DTS}=125\text{ns}$ (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 μs to 31750 ns by 250 ns steps,

32 μs to 63 μs by 1 μs steps,

64 μs to 126 μs by 2 μs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

22.5.16 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x48

Reset value: 0x0000

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

22.5.17 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

$$(\text{TIMx_CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4$$

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

22.5.18 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 124. TIM15 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIM15_CR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	U1FREMAP	Res	CKD [1:0]	ARPE	Res	Res	Res	Res	OPM	URS	UDIS	CFN
	Reset value																				0		0										
0x04	TIM15_CR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OIS2	OIS1N	OIS1	T1S	MMS[2:0]		CCDS	CCUS	Res	CCPC	
	Reset value																					0	0	0	0	0	0	0	0	0		0	
0x08	TIM15_SMCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	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
0x0C	TIM15_DIER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TDE	COMDE	Res	Res	Res	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res	Res	CC2IE	CC1IE
	Reset value																		0	0			0	0	0	0	0	0		0	0	0	
0x10	TIM15_SR	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	CC2IF	CC1IF	
	Reset value																					0	0		0	0	0			0	0	0	
0x14	TIM15_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	BG	TG	COMG	Res	Res	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	TIM15_CCMR1 Output Compare mode	Res	Res	Res	Res	Res	Res	Res	OC2M[3]	Res	Res	Res	Res	Res	Res	Res	OC1M[3]	Res	OC2M [2:0]		OC2PE	OC2FE	CC2S [1:0]		OC1M [2:0]		OC1PE		OC1FE	CC1S [1:0]			
	Reset value								0								0		0	0	0	0	0	0		0	0	0	0	0	0	0	
	TIM15_CCMR1 Input Capture mode	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	IC2F[3:0]		IC2PSC [1:0]	CC2S [1:0]		IC1F[3:0]		IC1PSC [1:0]		CC1S [1:0]						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
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	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 124. TIM15 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x24	TIM15_CNT	UIFCPY or 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	0	0	0			
0x28	TIM15_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	TIM15_ARR	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	0	0	0	0	0			
0x30	TIM15_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	TIM15_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	TIM15_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			
0x44	TIM15_BDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]	DT[7:0]										
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x48	TIM15_DCR	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	TIM15_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			

Refer to [Section 3.2.2 on page 48](#) for the register boundary addresses.

22.6 TIM16&TIM17 registers

Refer to [Section 2.1 on page 42](#) for a list of abbreviations used in register descriptions.

22.6.1 TIM16&TIM17 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	UIF REM- AP	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 (TIMx),

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.

22.6.2 TIM16&TIM17 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	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_BKR register).

Bit 8 OIS1: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

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

Bit 3 CCDS: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

Note: This bit acts only on channels that have a complementary output.

22.6.3 TIM16 &TIM17 DMA/interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	TDE	COMDE	Res	Res	Res	CC1DE	UDE	BIE	TIE	COMIE	Res	Res	Res	CC1IE	UIE
	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 12:10 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 4:2 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

22.6.4 TIM16&TIM17 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	CC1OF	Res	BIF	TIF	COMIF	Res	Res	Res	CC1IF	UIF
						rc_w0		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 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software.

0: No trigger event occurred

1: Trigger interrupt pending

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

If channel CC1 is configured as output:

This flag is set by hardware when the counter matches the compare value. It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow

If channel CC1 is configured as input:

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow 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 22.5.3: TIM15 slave mode control register \(TIM15_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx_CR1 register.

22.6.5 TIM16&TIM17 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	BG	TG	COMG	Res	Res	Res	CC1G	UG
								w	w	w				w	w

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

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When 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).

22.6.6 TIM16&TIM17 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	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]
															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	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								IC1F[3:0]			IC1PSC[1:0]				
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:17 Reserved, always read as 0

Bit 16 **OC1M[3]**: Output Compare 1 mode (bit 3)

Bits 15:7 Reserved

Bits 6:4 **OC1M[2:0]**: Output Compare 1 mode (bits 2 to 0)

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.

All other values: Reserved

Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

2: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

2: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

Bits 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 events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).

The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input.

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

22.6.7 TIM16&TIM17 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

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

2. 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

CC1 channel configured as output:

- 0: OC1 active high
- 1: OC1 active low

CC1 channel configured as input:

The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

00: 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).

01: 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).

10: Reserved, do not use this configuration.

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).

Note: 1. This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

2. 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

CC1 channel configured as output:

- 0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
- 1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input:

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

- 0: Capture disabled
- 1: Capture enabled

Table 125. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states ⁽¹⁾	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	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) OCx=CCxP, OCxN=CCxNP	
	1		0	0		
			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 AFIO registers.*

22.6.8 TIM16&TIM17 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

22.6.9 TIM16&TIM17 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to $f_{CK_PSC} / (PSC[15:0] + 1)$.

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

22.6.10 TIM16&TIM17 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 22.4.1: Time-base unit on page 620](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

22.6.11 TIM16&TIM17 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000

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

22.6.12 TIM16&TIM17 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
r/w	r/w	r/w	r/w	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).

22.6.13 TIM16&TIM17 break and dead-time register (TIMx_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	Res	Res	Res	Res	Res	Res	Res	Res	Res	BKF[3:0]			
												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 BKF[3:0], AOE, BKP, BKE, OSSR, OSSI 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 31:20 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 22.5.8: TIM15 capture/compare enable register \(TIM15_CCER\) on page 662](#)).

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: 1. This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

2. 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: 1. This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

2. 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 22.5.8: TIM15 capture/compare enable register \(TIM15_CCER\) on page 662](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO 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 on channels configured as outputs.

See OC/OCN enable description for more details ([Section 22.5.8: TIM15 capture/compare enable register \(TIM15_CCER\) on page 662](#)).

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 => $DT = DTG[7:0] \times t_{dtg}$ with $t_{dtg} = t_{DTS}$

DTG[7:5]=10x => $DT = (64 + DTG[5:0]) \times t_{dtg}$ with $t_{dtg} = 2 \times t_{DTS}$

DTG[7:5]=110 => $DT = (32 + DTG[4:0]) \times t_{dtg}$ with $t_{dtg} = 8 \times t_{DTS}$

DTG[7:5]=111 => $DT = (32 + DTG[4:0]) \times t_{dtg}$ with $t_{dtg} = 16 \times t_{DTS}$

Example if $T_{DTS} = 125\text{ns}$ (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 μs to 31750 ns by 250 ns steps,

32 μs to 63 μs by 1 μs steps,

64 μs to 126 μs by 2 μs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

22.6.14 TIM16&TIM17 DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

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

22.6.15 TIM16&TIM17 DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write 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).

22.6.16 TIM16 option register (TIM16_OR)

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	TI1RMP	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits1:0 **TI1_RMP**: Timer 16 input 1 connection.

This bit is set and cleared by software.

00: TIM16 TI1 is connected to GPIO

01: TIM16 TI1 is connected to RTC_clock

10: TIM16 TI1 is connected to HSE/32

11: TIM16 TI1 is connected to MCO

22.6.17 TIM16&TIM17 register map

TIM16 & TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 126. TIM16&TIM17 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	TIMx_CR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UJFREMAP	Res	CKD [1:0]	0	ARPE	Res	Res	Res	OPM	URS	UDIS	CEN	
	Reset value																					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				0	0		0	CCPC	
0x0C	TIMx_DIER	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	CC1IE	UIE	
	Reset value																		0	0	0	0	0	0	0	0	0				0	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	CC1OF	Res	BIF	TIF	COMIF	Res	Res	Res	CC1IF	UIF	
	Reset value																						0		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	BG	TG	COMG	Res	Res	Res	CC1G	UG	
	Reset value																									0	0	0				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	OC1M [2:0]			OC1PE	OC1FE	CC1s [1:0]		
	Reset value																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	IC1F[3:0]			IC1PSC [1:0]	CC1s [1:0]				
	Reset value																									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	CC1NP	CC1NE	CC1P	CC1E	
	Reset value																												0	0	0	0	0	
0x24	TIMx_CNT	UIFCPY or 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	0	0	0	0	

Table 126. TIM16&TIM17 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	TIMx_PSC	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	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	0	0	0	0	0	0
0x30	TIMx_RCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	REP[7:0]						
	Reset value																									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	Res	Res	Res	Res	Res	Res	Res	Res	Res	BKF[3:0]			MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]	DT[7:0]									
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	DBL[4:0]				Res	Res	Res	DBA[4:0]					
	Reset value																				0	0	0	0	0				0	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	TIM16_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	T11 RMP [1:0]
	Reset value																																0

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

23 Basic timers (TIM6)

23.1 TIM6 introduction

The basic timer TIM6 consists 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.

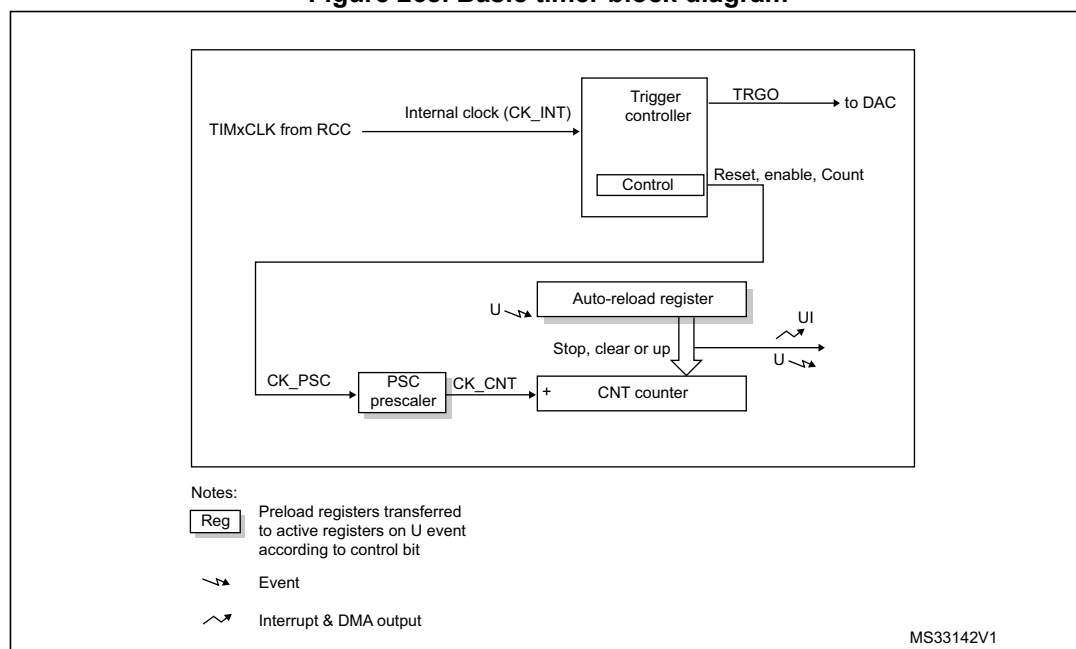
The timers are completely independent, and do not share any resources.

23.2 TIM6 main features

Basic timer (TIM6) 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 268. Basic timer block diagram



23.3 TIM6 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 269 and *Figure 270* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 269. Counter timing diagram with prescaler division change from 1 to 2

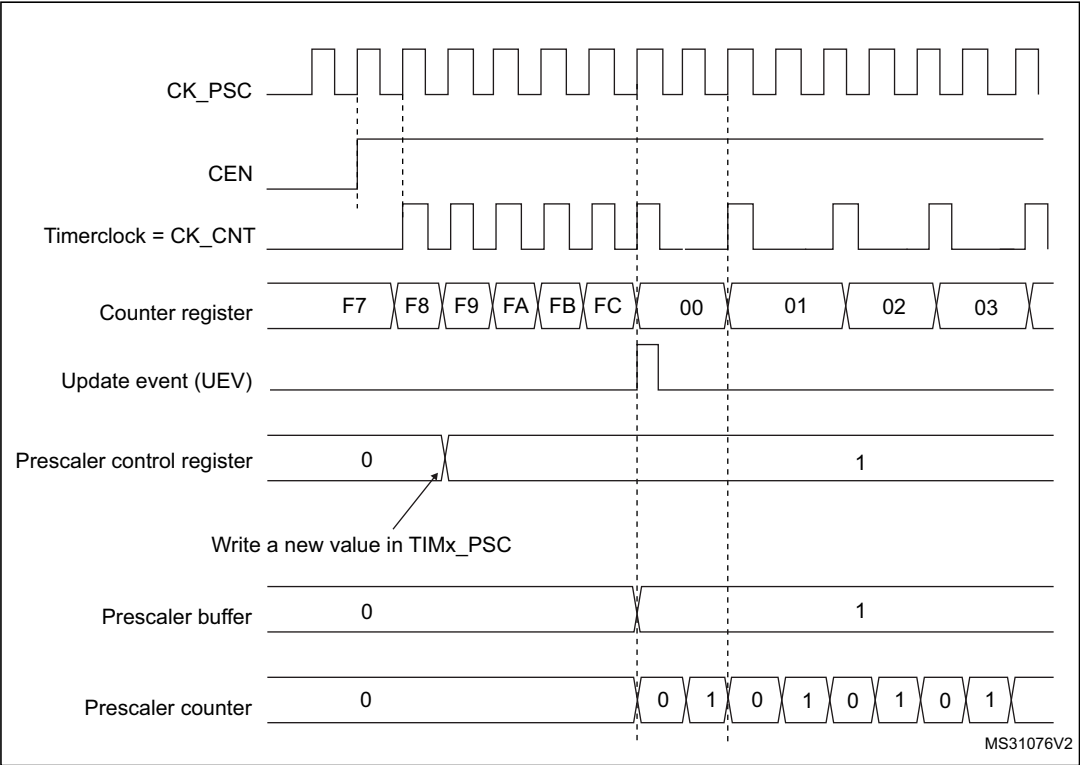
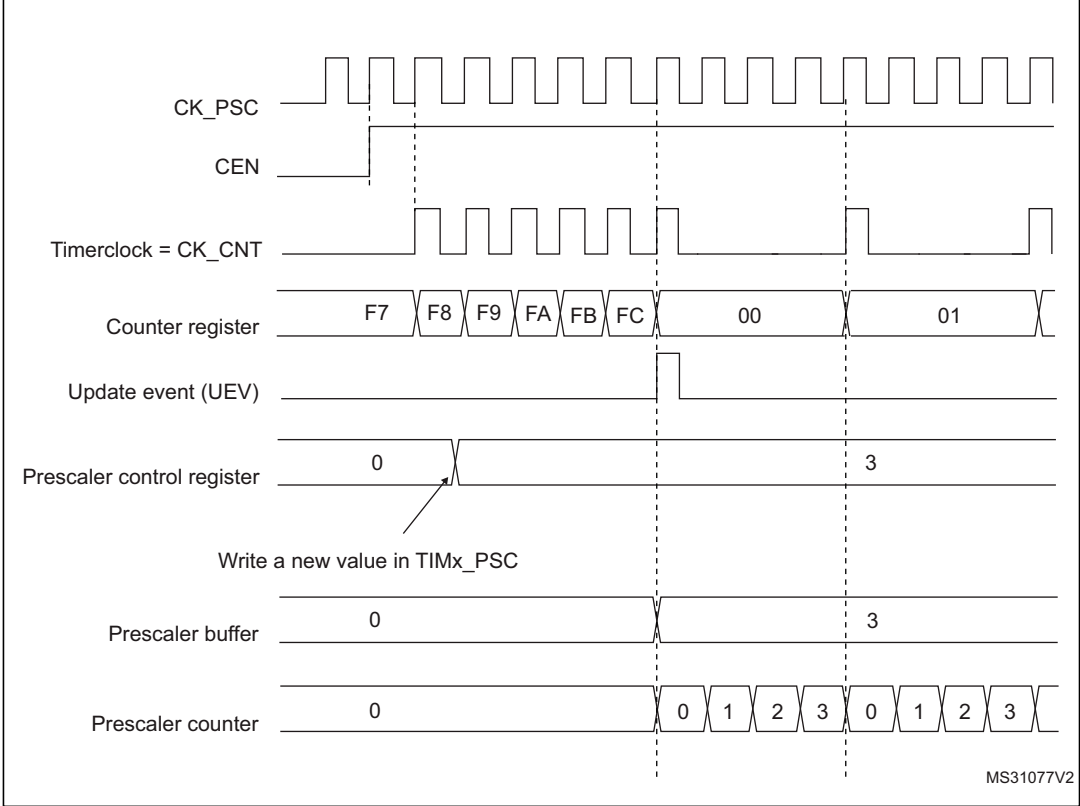


Figure 270. Counter timing diagram with prescaler division change from 1 to 4



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 written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 271. Counter timing diagram, internal clock divided by 1

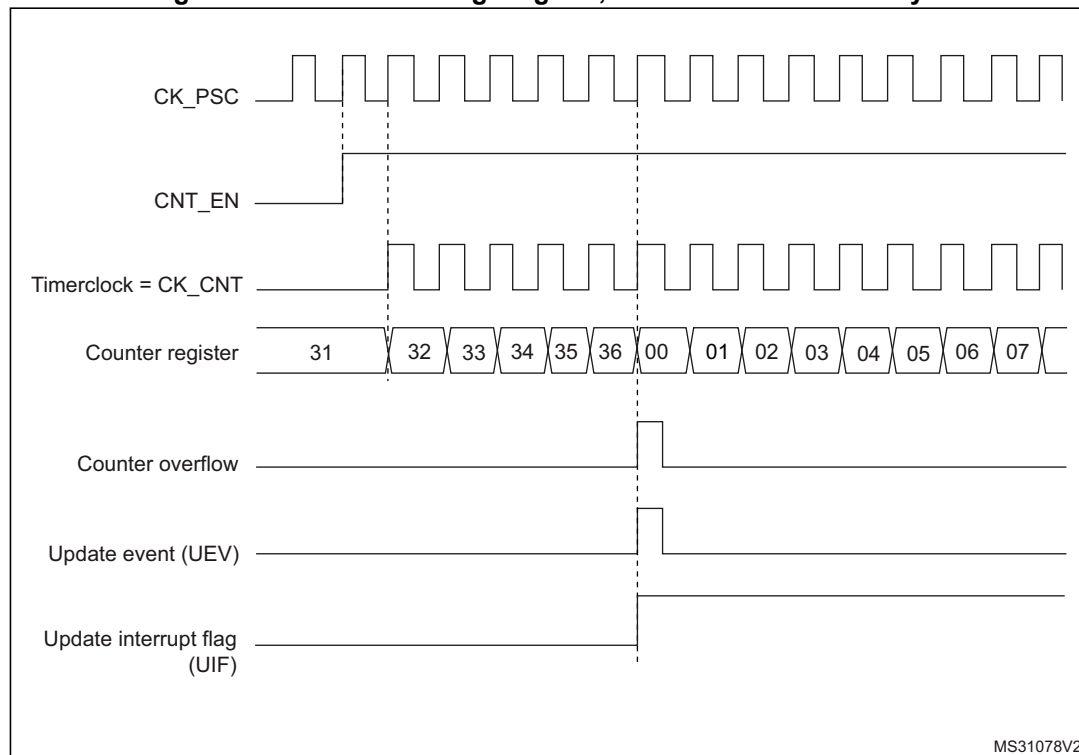


Figure 272. Counter timing diagram, internal clock divided by 2

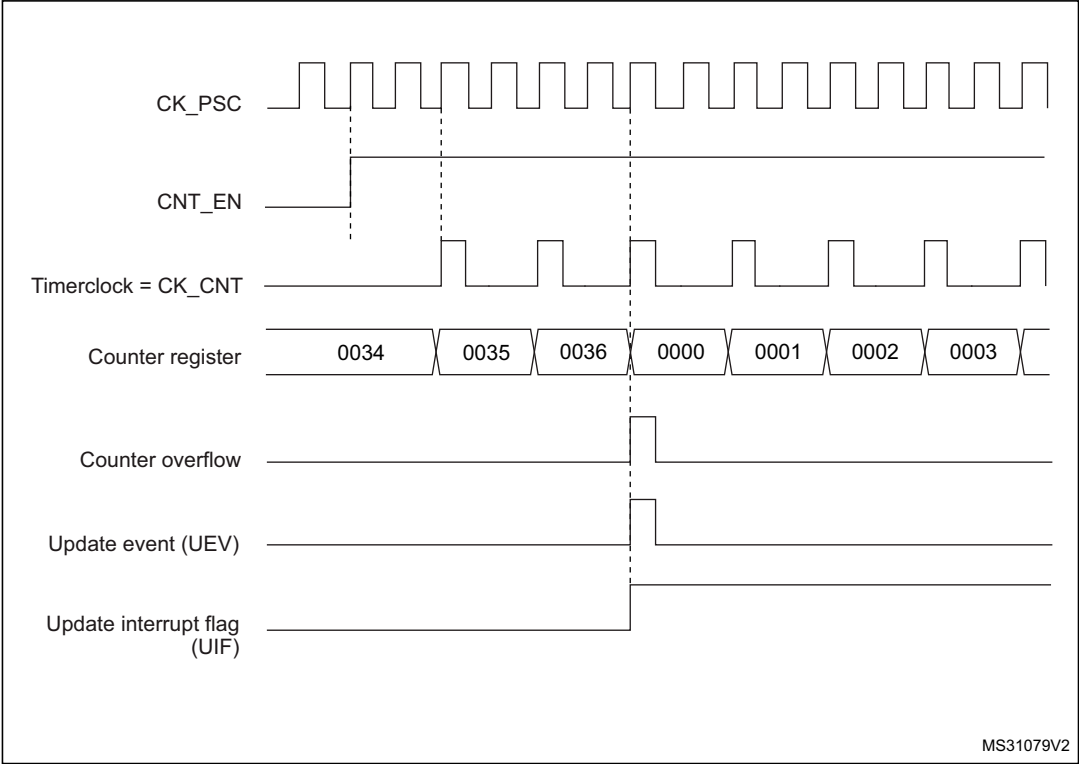


Figure 273. Counter timing diagram, internal clock divided by 4

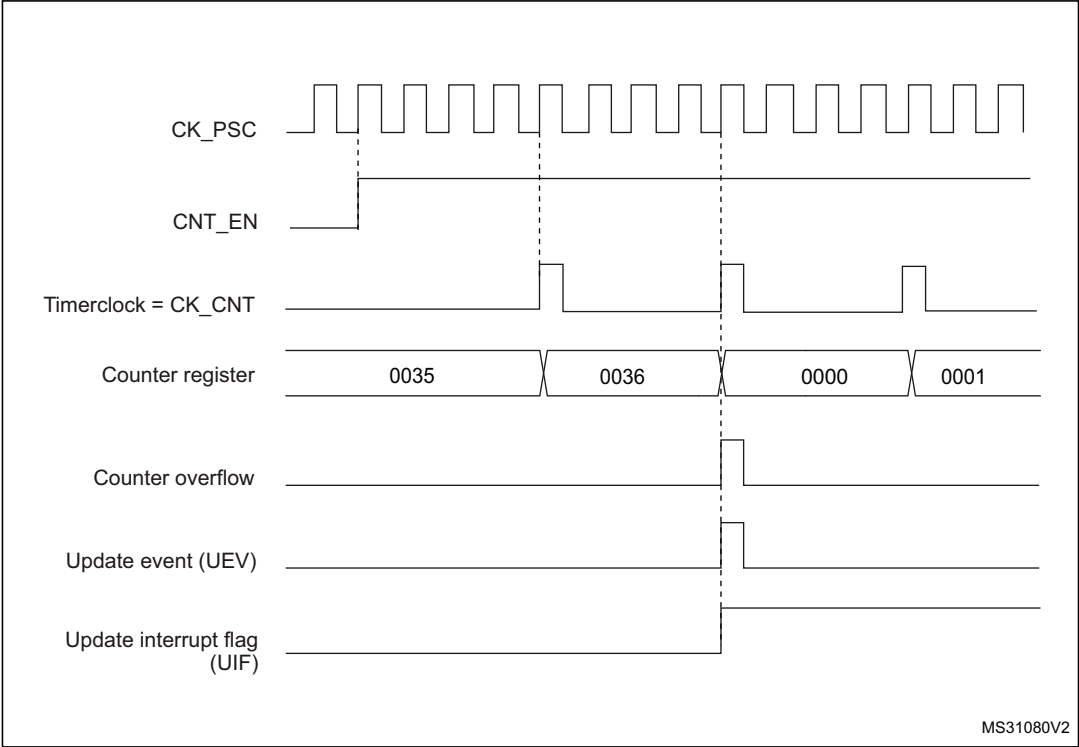


Figure 274. Counter timing diagram, internal clock divided by N

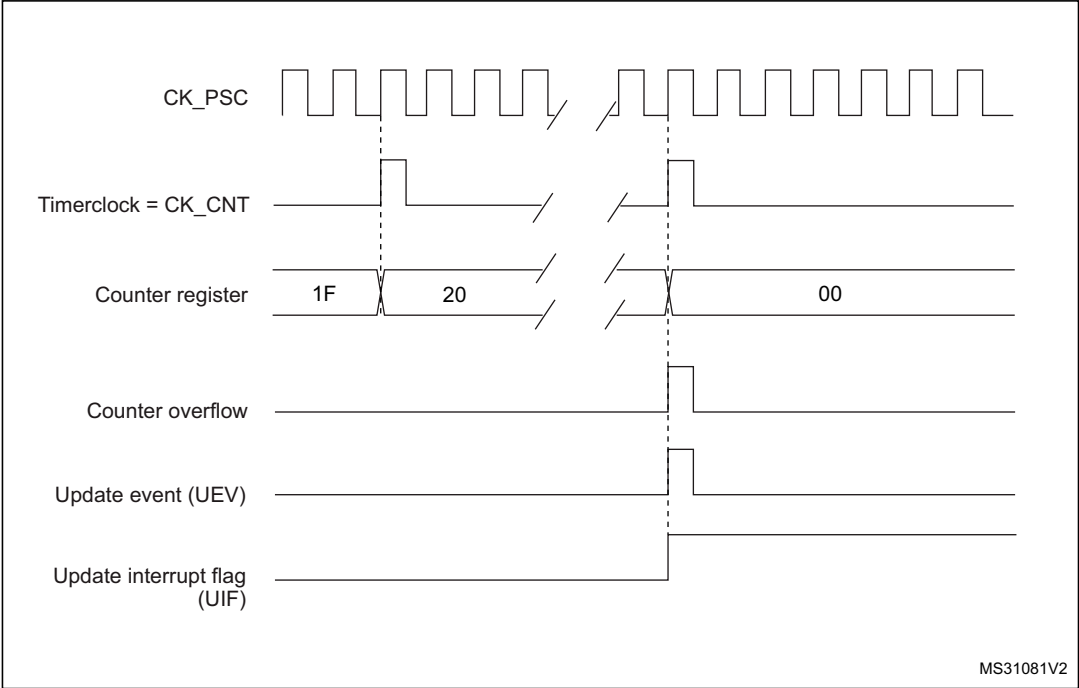


Figure 275. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

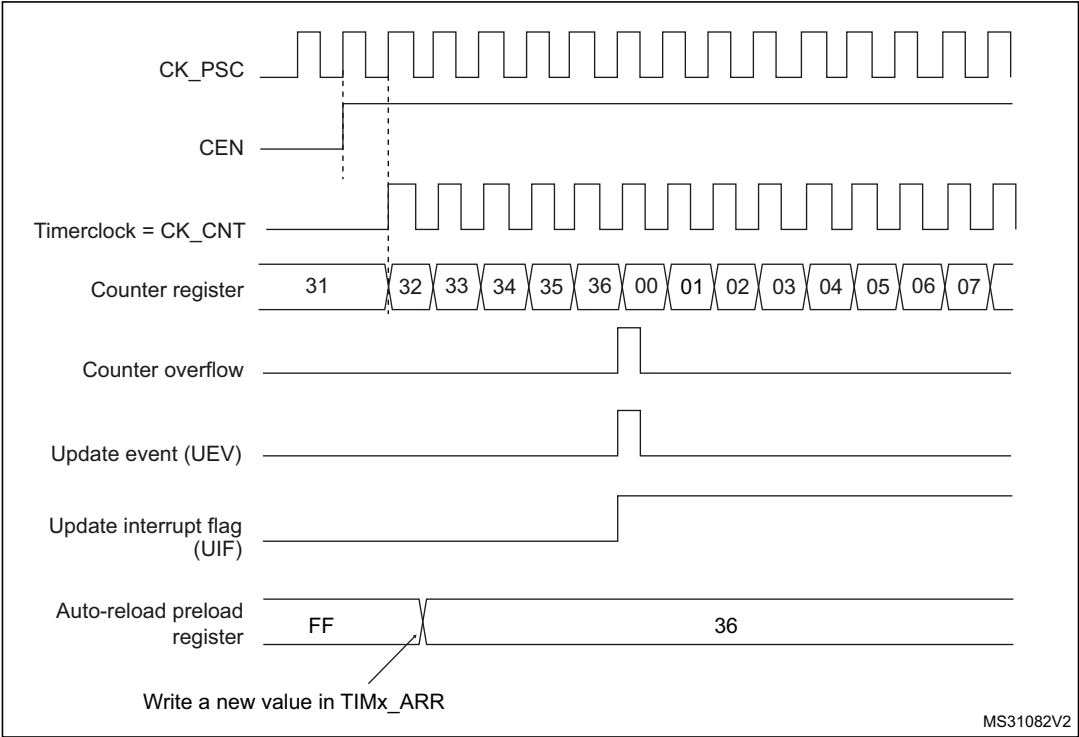
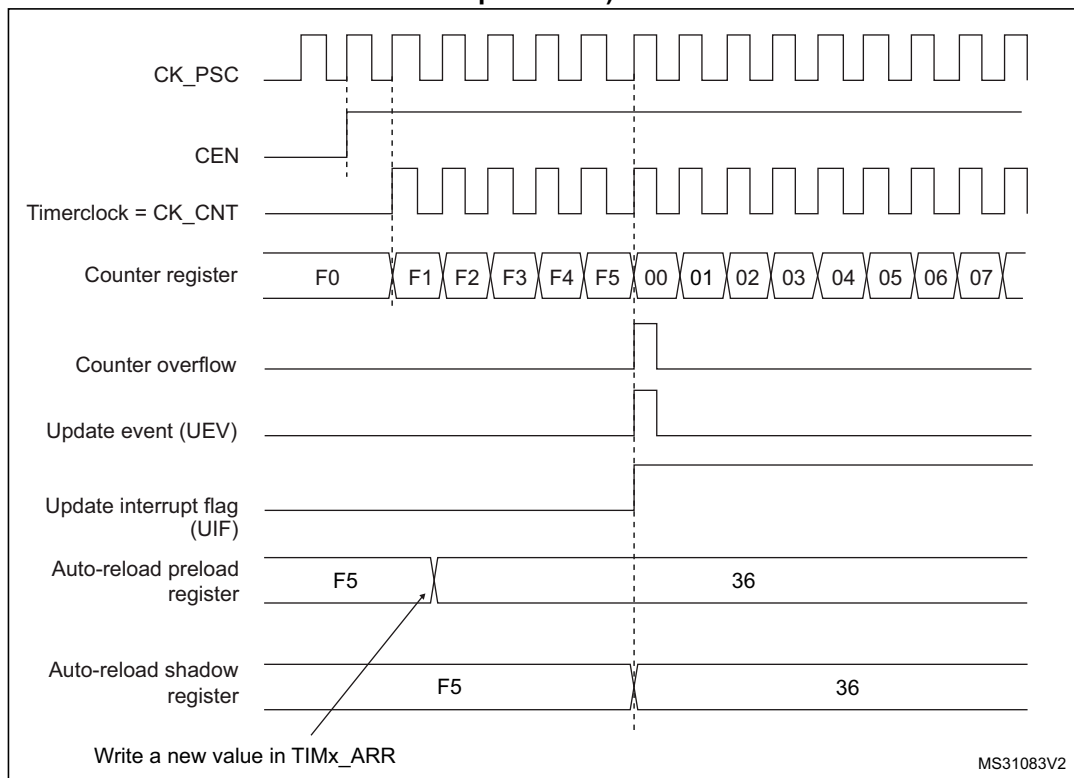


Figure 276. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



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.

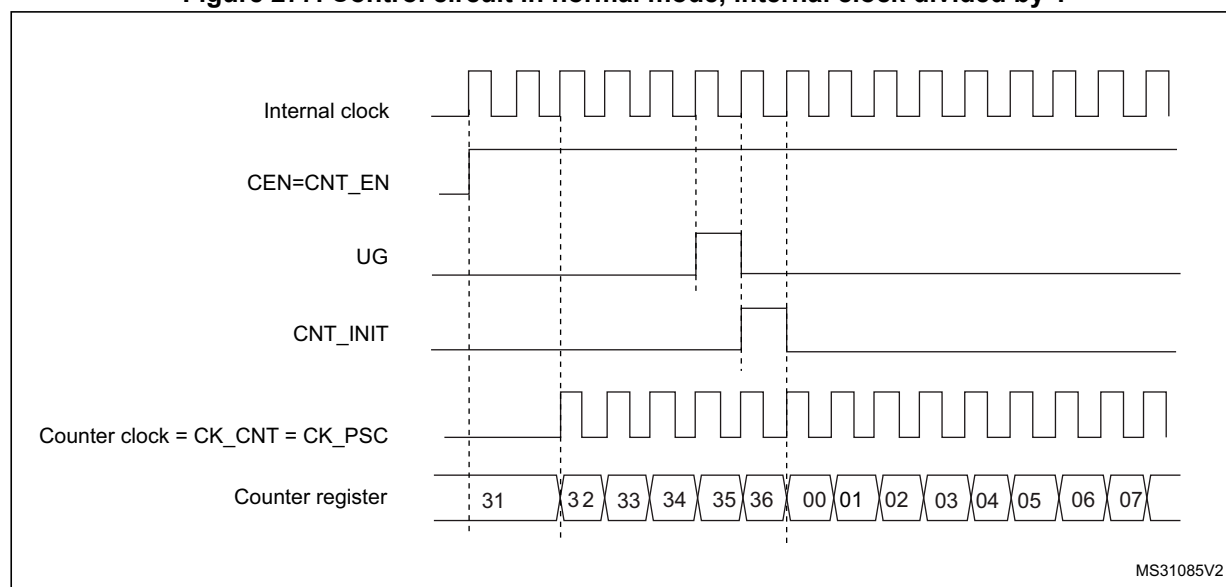
23.3.4 Clock source

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 277 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 277. Control circuit in normal mode, internal clock divided by 1



23.3.5 Debug mode

When the microcontroller enters the debug mode (Cortex-M4[®]F 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 33.14.2: Debug support for timers, watchdog, bxCAN and I2C](#).

23.4 TIM6 registers

Refer to [Section 2.1 on page 42](#) 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 TIM6 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	UIF RE-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.

Bit 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 TIM6 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	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**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).

010: **Update** - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

Bits 3:0 Reserved, must be kept at reset value.

23.4.3 TIM6 DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	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 TIM6 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	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 TIM6 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	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 TIM6 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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 TIM6 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to $f_{CK_PSC} / (PSC[15:0] + 1)$.

PSC contains the value to be loaded into the active prescaler register at each update event.

23.4.8 TIM6 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded into the actual auto-reload register.

Refer to [Section 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 TIM6 register map

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

Table 127. TIM6 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
0x00	TIMx_CR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	U/IFREMAP	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	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	UDE	Res	Res	Res	Res	Res	Res	Res	UIE						
	Reset value																								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	Res	Res	Res	Res	Res	UIF						
	Reset value																																	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	UG						
	Reset value																																	0					
0x18-0x20	Reserved																																						
0x24	TIMx_CNT	U/IFCOPY or 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	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	PSC[15:0]															
	Reset value																	0	0	0	0	0	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	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

24 Infrared interface (IRTIM)

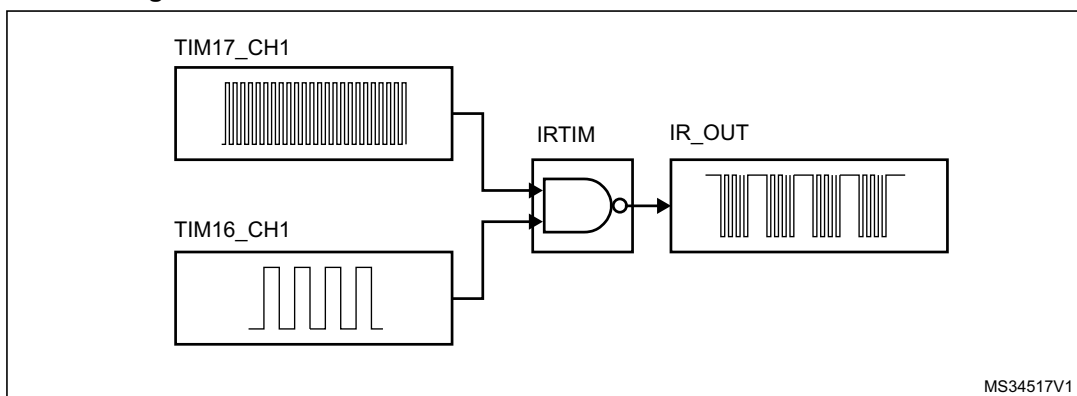
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with TIM16 and TIM17 as shown in [Figure 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. IR internal hardware connections with TIM16 and TIM17



All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 generates the modulation envelope.

The infrared function is output on the IR_OUT pin. The activation of this function is done through the GPIOx_AFRx register by enabling the related alternate function bit.

The high sink LED driver capability (only available on the PB9 pin) can be activated through the I2C_PB9_FMP bit in the SYSCFG_CFGR1 register and used to sink the high current needed to directly control an infrared LED.

25 System window watchdog (WWDG)

25.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB1 clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications which require the watchdog to react within an accurate timing window.

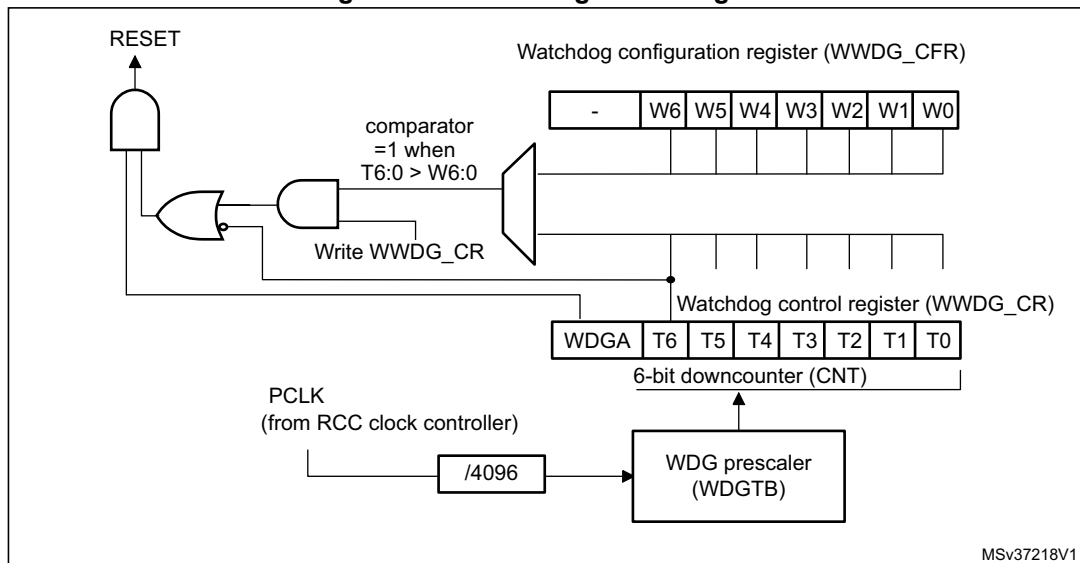
25.2 WWDG main features

- Programmable free-running downcounter
- Conditional reset
 - Reset (if watchdog activated) when the downcounter value becomes less than 0x40
 - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see [Figure 280](#))
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

25.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit downcounter (T[6:0] bits) rolls over from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

Figure 279. Watchdog block diagram



The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

25.3.1 Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

25.3.2 Controlling the downcounter

This downcounter 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 which represents the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register (see [Figure 280](#)). The Configuration register (WWDG_CFR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x3F. [Figure 280](#) 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).

25.3.3 Advanced watchdog interrupt feature

The Early Wakeup Interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) should reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

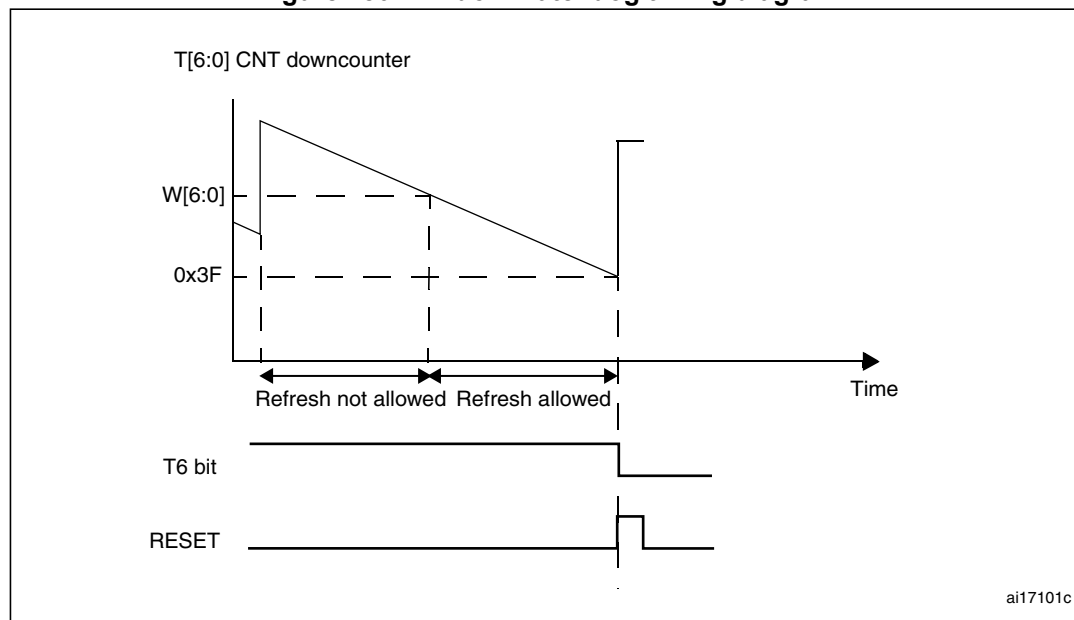
Note: When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset will eventually be generated.

25.3.4 How to program the watchdog timeout

You can use the formula in [Figure 280](#) 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 280. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{\text{WWDG}} = t_{\text{PCLK1}} \times 4096 \times 2^{\text{WDGTB}[1:0]} \times (T[5:0] + 1) \quad (\text{ms})$$

where:

t_{WWDG} : WWDG timeout

t_{PCLK1} : APB1 clock period measured in ms

4096: value corresponding to internal divider

As an example, let's assume APB1 frequency is equal to 48 MHz, WDGTB[1:0] is set to 3 and T[5:0] is set to 63:

$$t_{\text{WWDG}} = 1/48000 \times 4096 \times 2^3 \times (63 + 1) = 43.69 \text{ ms}$$

Refer to the datasheet for the minimum and maximum values of the t_{WWDG} .

25.3.5 Debug mode

When the microcontroller enters debug mode (Cortex-M4[®]F core halted), the WWDG counter either continues to work normally or stops, depending on DBG_WWDG_STOP configuration bit in DBG module. For more details, refer to .

25.4 WWDG registers

Refer to [Section 2.1 on page 42](#) for a list of abbreviations used in register descriptions.

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

25.4.1 Control register (WWDG_CR)

Address offset: 0x00

Reset value: 0x0000 007F

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

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

Bit 7 **WDGA**: Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]**: 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter. It is decremented every $(4096 \times 2^{\text{WDGTB}[1:0]})$ PCLK cycles. A reset is produced when it rolls over from 0x40 to 0x3F (T6 becomes cleared).

25.4.2 Configuration register (WWDG_CFR)

Address offset: 0x04

Reset value: 0x0000 007F

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

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

Bit 9 **EWI**: Early wakeup interrupt

When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 **WDGTB[1:0]**: Timer base

The time base of the prescaler can be modified as follows:

- 00: CK Counter Clock (PCLK div 4096) div 1
- 01: CK Counter Clock (PCLK div 4096) div 2
- 10: CK Counter Clock (PCLK div 4096) div 4
- 11: CK Counter Clock (PCLK div 4096) div 8

Bits 6:0 **W[6:0]**: 7-bit window value

These bits contain the window value to be compared to the downcounter.

25.4.3 Status register (WWDG_SR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bit 0 **EWIF**: Early wakeup interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing '0'. A write of '1' has no effect. This bit is also set if the interrupt is not enabled.

25.4.4 WWDG register map

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

Table 128. WWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0x00	WWDG_CR	Res.	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				
0x04	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	WDGTB1	WDGTB0	W[6:0]										
	Reset value																							0	0	0	1	1	1	1	1	1	1				
0x08	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																								0	0	0	0	0	0	0	0	0				

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

26 Independent watchdog (IWDG)

26.1 Introduction

The devices feature an embedded watchdog peripheral which offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral serves to detect and resolve malfunctions due to software failure, and to trigger system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to [Section 25 on page 708](#).

26.2 IWDG main features

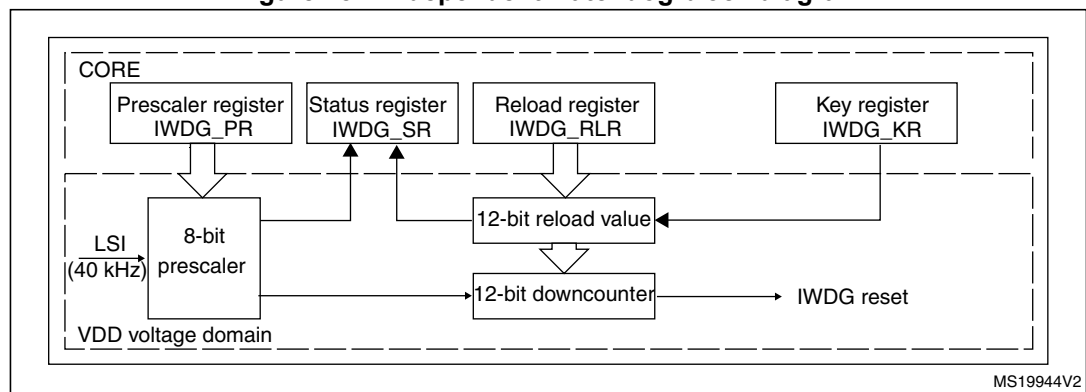
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional Reset
 - Reset (if watchdog activated) when the downcounter value becomes less than 0x000
 - Reset (if watchdog activated) if the downcounter is reloaded outside the window

26.3 IWDG functional description

26.3.1 IWDG block diagram

[Figure 281](#) shows the functional blocks of the independent watchdog module.

Figure 281. Independent watchdog block diagram



Note: The watchdog function is implemented in the CORE voltage domain that is still functional in Stop and Standby modes.

When the independent watchdog is started by writing the value 0x0000 CCCC in the Key register (IWDG_KR), the counter starts counting down from the reset value of 0xFFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

26.3.2 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the IWDG_WINR register.

If the reload operation is performed while the counter is greater than the value stored in the window register (IWDG_WINR), then a reset is provided.

The default value of the IWDG_WINR is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the IWDG_RLR value and ease the cycle number calculation to generate the next reload.

Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the IWDG_KR register.
2. Enable register access by writing 0x0000 5555 in the IWDG_KR register.
3. Write the IWDG prescaler by programming IWDG_PR from 0 to 7.
4. Write the reload register (IWDG_RLR).
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Write to the window register IWDG_WINR. This automatically refreshes the counter value IWDG_RLR.

Note: Writing the window value allows to refresh the Counter value by the RLR when IWDG_SR is set to 0x0000 0000.

Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the IWDG_KR register.
2. Enable register access by writing 0x0000 5555 in the IWDG_KR register.
3. Write the IWDG prescaler by programming IWDG_PR from 0 to 7.
4. Write the reload register (IWDG_RLR).
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Refresh the counter value with IWDG_RLR (IWDG_KR = 0x0000 AAAA)

26.3.3 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the Key register is written by the software before the counter reaches end of count or if the downcounter is reloaded inside the window.

26.3.4 Register access protection

Write access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers is protected. To modify them, you must first write the code 0x0000 5555 in the IWDG_KR register. A write access to this register with a different value will break the sequence and register access will be protected again. This implies that it is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is on going.

26.3.5 Debug mode

When the microcontroller enters debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on DBG_IWDG_STOP configuration bit in DBG module.

26.4 IWDG registers

Refer to [Section 2.1 on page 42](#) for a list of abbreviations used in register descriptions.

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

26.4.1 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 26.3.4: Register access protection](#))

Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

26.4.2 Prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 26.3.4: Register access protection](#). They are written by software to select the prescaler divider feeding the counter clock. PVU bit of IWDG_SR must be reset in order to be able to change the prescaler divider.

000: divider /4
 001: divider /8
 010: divider /16
 011: divider /32
 100: divider /64
 101: divider /128
 110: divider /256
 111: divider /256

Note: Reading this register returns the prescaler value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG_SR register is reset.

26.4.3 Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

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

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

Bits11:0 **RL[11:0]**: Watchdog counter reload value

These bits are write access protected see [Section 26.3.4](#). They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the IWDG_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to the datasheet for the timeout information.

The RVU bit in the IWDG_SR register must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on this register. For this reason the value read from this register is valid only when the RVU bit in the IWDG_SR register is reset.

26.4.4 Status register (IWDG_SR)

Address offset: 0x0C

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1

Bit 1 **RVU**: Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU**: Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Prescaler value can be updated only when PVU bit is reset.

Note: *If several reload, 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.*

26.4.5 Window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF (reset by Standby mode)

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

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

Bits 11:0 **WIN[11:0]**: Watchdog counter window value

These bits are write access protected see [Section 26.3.4](#). These bits contain the high limit of the window value to be compared to the downcounter.

To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0

The WVU bit in the IWDG_SR register must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG_SR register is reset.

26.4.6 IWDG register map

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

Table 129. IWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	IWDG_KR	Res	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	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	RL[11:0]												
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1	
0x0C	IWDG_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WVU	RVU	PVU
	Reset value																														0	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	WIN[11:0]												
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1	

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

27 Real-time clock (RTC)

27.1 Introduction

The RTC provides an automatic wakeup to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

The RTC includes also a periodic programmable wakeup flag with interrupt capability.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.

Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After Backup domain reset, all RTC registers are protected against possible parasitic write accesses.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

27.2 RTC main features

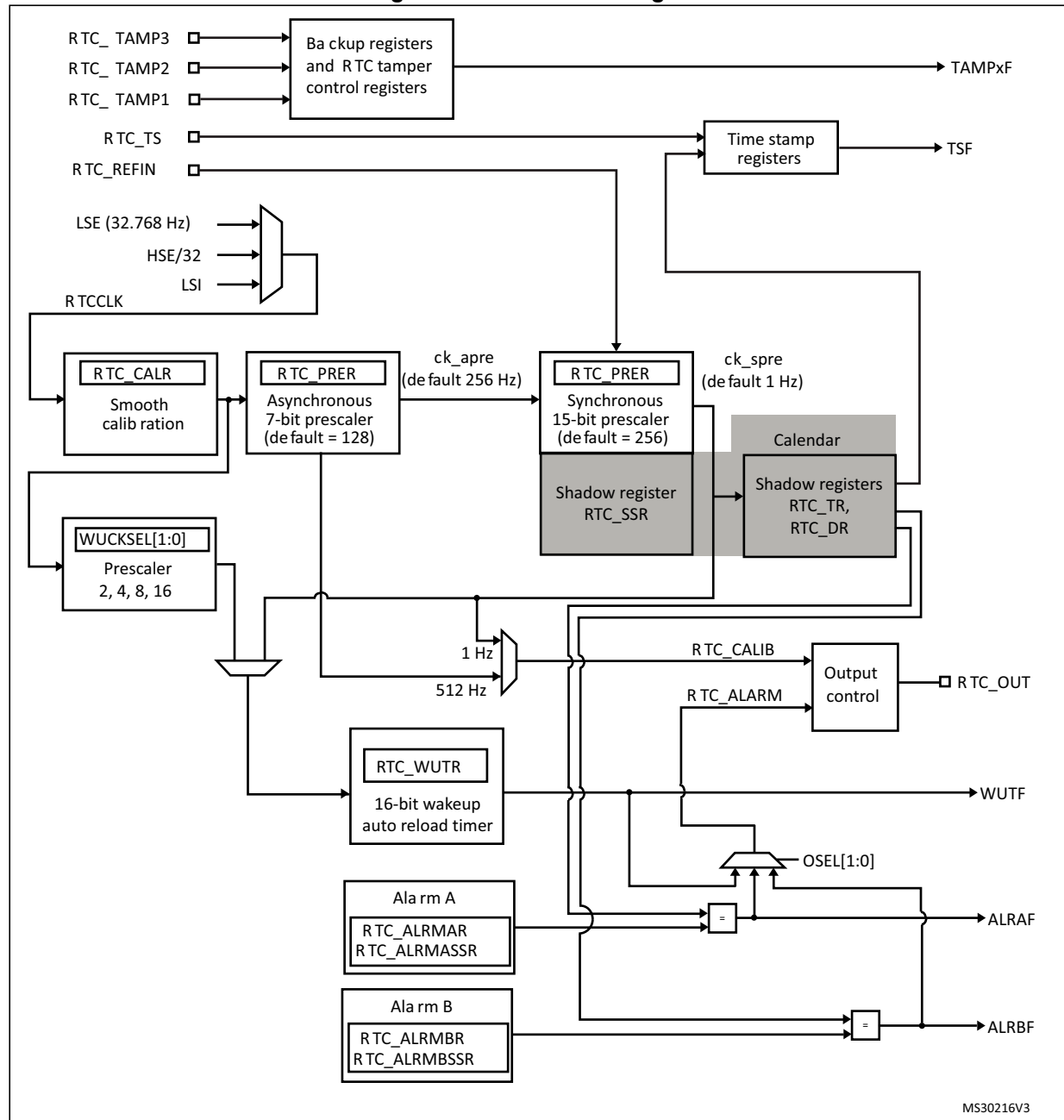
The RTC unit main features are the following (see [Figure 282: RTC block diagram](#)):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Programmable alarm with interrupt function. The alarm can be triggered by any combination of the calendar fields.
- Automatic wakeup unit generating a periodic flag that triggers an automatic wakeup interrupt.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Accurate synchronization with an external clock using the subsecond shift feature.
- Digital calibration circuit (periodic counter correction): 0.95 ppm accuracy, obtained in a calibration window of several seconds
- Time-stamp function for event saving
- Tamper detection event with configurable filter and internal pull-up
- Maskable interrupts/events:
 - Alarm A
 - Alarm B
 - Wakeup interrupt
 - Time-stamp
 - Tamper detection
- Backup registers.

27.3 RTC functional description

27.3.1 RTC block diagram

Figure 282. RTC block diagram



The RTC includes:

- Two alarms
- Three tamper events
- 16 x 32-bit backup registers in STM32F302xB/C and 5 x 32-bit backup registers in STM32F302x6/8
 - The backup registers (RTC_BKPxR) are implemented in the RTC domain that remains powered-on by VBAT when the VDD power is switched off.
- Alternate function outputs: RTC_OUT which selects one of the following two outputs:
 - RTC_CALIB: 512 Hz or 1Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
 - RTC_ALARM: This output is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the Alarm A, Alarm B or Wakeup outputs.
- Alternate function inputs:
 - RTC_TS: timestamp event
 - RTC_TAMP1: tamper1 event detection
 - RTC_TAMP2: tamper2 event detection
 - RTC_TAMP3: tamper3 event detection
 - RTC_REFIN: 50 or 60 Hz reference clock input

27.3.2 GPIOs controlled by the RTC

RTC_OUT, RTC_TS and RTC_TAMP1 are mapped on the same pin (PC13).

The selection of the RTC_ALARM output is performed through the RTC_TAFCR register as follows: the PC13VALUE bit is used to select whether the RTC_ALARM output is configured in push-pull or open drain mode.

When PC13 is not used as RTC alternate function, it can be forced in output push-pull mode by setting the PC13MODE bit in the RTC_TAFCR. The output data value is then given by the PC13VALUE bit. In this case, PC13 output push-pull state and data are preserved in Standby mode.

The output mechanism follows the priority order shown in [Table 130](#).

When PC14 and PC15 are not used as LSE oscillator, they can be forced in output push-pull mode by setting the PC14MODE and PC15MODE bits in the RTC_TAFCR register respectively. The output data values are then given by PC14VALUE and PC15VALUE. In this case, the PC14 and PC15 output push-pull states and data values are preserved in Standby mode.

The output mechanism follows the priority order shown in [Table 131](#) and [Table 132](#).

Table 130. RTC pin PC13 configuration⁽¹⁾

Pin configuration and function	RTC_ALARM output enabled	RTC_CALIB output enabled	RTC_TAMP1 input enabled	RTC_TS input enabled	PC13MODE bit	PC13VALUE bit
RTC_ALARM output OD	1	Don't care	Don't care	Don't care	Don't care	0
RTC_ALARM output PP	1	Don't care	Don't care	Don't care	Don't care	1

Table 130. RTC pin PC13 configuration⁽¹⁾ (continued)

Pin configuration and function	RTC_ALARM output enabled	RTC_CALIB output enabled	RTC_TAMP1 input enabled	RTC_TS input enabled	PC13MODE bit	PC13VALUE bit
RTC_CALIB output PP	0	1	Don't care	Don't care	Don't care	Don't care
RTC_TAMP1 input floating	0	0	1	0	Don't care	Don't care
RTC_TS and RTC_TAMP1 input floating	0	0	1	1	Don't care	Don't care
RTC_TS input floating	0	0	0	1	Don't care	Don't care
Output PP forced	0	0	0	0	1	PC13 output data value
Wakeup pin or Standard GPIO	0	0	0	0	0	Don't care

1. OD: open drain; PP: push-pull.

Table 131. LSE pin PC14 configuration ⁽¹⁾

Pin configuration and function	LSEON bit in RCC_BDCR register	LSEBYP bit in RCC_BDCR register	PC14MODE bit	PC14VALUE bit
LSE oscillator	1	0	Don't care	Don't care
LSE bypass	1	1	Don't care	Don't care
Output PP forced	0	Don't care	1	PC14 output data value
Standard GPIO	0	Don't care	0	Don't care

1. OD: open drain; PP: push-pull.

Table 132. LSE pin PC15 configuration ⁽¹⁾

Pin configuration and function	LSEON bit in RCC_BDCR register	LSEBYP bit in RCC_BDCR register	PC15MODE bit	PC15VALUE bit
LSE oscillator	1	0	Don't care	Don't care
Output PP forced	1	1	1	PC15 output data value
	0	Don't care		
Standard GPIO	0	Don't care	0	Don't care

1. OD: open drain; PP: push-pull.

27.3.3 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to [Section 10: 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 wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see [Section 27.3.6: Periodic auto-wakeup](#) for details).

27.3.4 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every two RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ISR register is set (see [Section 27.6.4: RTC initialization and status](#)

register (RTC_ISR)). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 2 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD=0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

27.3.5 Programmable alarms

The RTC unit provides programmable alarm: Alarm A and Alarm B. The description below is given for Alarm A, but can be translated in the same way for Alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register. The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASR register. The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and Alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the RTC_ALARM output. RTC_ALARM output polarity can be configured through bit POL the RTC_CR register.

27.3.6 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC_CR register.

The wakeup timer clock input can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.
When RTCCLK is LSE(32.768kHz), this allows to configure the wakeup interrupt period from 122 μ s to 32 s, with a resolution down to 61 μ s.
- ck_spre (usually 1 Hz internal clock)
When ck_spre frequency is 1Hz, this allows to achieve a wakeup time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
 - from 1s to 18 hours when WUCKSEL [2:1] = 10
 - and from around 18h to 36h when WUCKSEL[2:1] = 11. In this last case 2¹⁶ is added to the 16-bit counter current value. When the initialization sequence is complete (see [Programming the wakeup timer on page 730](#)), the timer starts counting down. When the wakeup function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in

the RTC_ISR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR2 register, it can exit the device from low-power modes.

The periodic wakeup flag can be routed to the RTC_ALARM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. RTC_ALARM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wakeup timer.

27.3.7 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by clearing the DBP bit in the PWR_CR register (refer to the power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, all the RTC registers are write-protected. Writing to the RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.

The following steps are required to unlock the write protection on all the RTC registers except for RTC_TAFCR, RTC_BKPxR and RTC_ISR[13:8].

1. Write '0xCA' into the RTC_WPR register.
2. Write '0x53' into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ISR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ISR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

Note: *After a system reset, the application can read the INITS flag in the RTC_ISR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00). To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ISR register.*

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for Alarm A but can be translated in the same way for Alarm B.

1. Clear ALRAE in RTC_CR to disable Alarm A.
2. Program the Alarm A registers (RTC_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 wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ISR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wakeup timer restarts down-counting.

27.3.8 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third

read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ISR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every two RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 2 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to [Calendar initialization and configuration on page 729](#)): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to [Section 27.3.10: RTC synchronization](#)): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (STOP or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD=1, instructions which read the calendar registers require one extra APB cycle to complete.

27.3.9 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ISR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register

(RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the RTC tamper and alternate function configuration register (RTC_TAFCR), the RTC backup registers (RTC_BKPxR), the wakeup timer register (RTC_WUTR), the Alarm A and Alarm B registers (RTC_ALRMASR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR).

In addition, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one. When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

27.3.10 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

RTC_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of $1 / (\text{PREDIV}_S + 1)$ seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 μ 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.

27.3.11 RTC reference clock detection

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_apre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREDIV_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

27.3.12 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about 2^{20} RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, cal_cnt[19:0], clocked by RTCCLK.

The smooth calibration register (RTC_CALR) specifies the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of RTCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to '1' causes exactly one pulse to be masked during the 32-second cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1]=1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2]=1

causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8]=1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to '1' effectively inserts an extra RTCCLK pulse every 2^{11} RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8- second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ISR/INITF=0, by using the follow process:

1. Poll the RTC_ISR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

27.3.13 Time-stamp function

Time-stamp is enabled by setting the TSE bit of RTC_CR register to 1.

The calendar is saved in the time-stamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a time-stamp event is detected on the RTC_TS pin.

When a time-stamp event occurs, the time-stamp flag bit (TSF) in RTC_ISR register is set.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a time-stamp event occurs.

If a new time-stamp event is detected while the time-stamp flag (TSF) is already set, the time-stamp overflow flag (TSOVF) flag is set and the time-stamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: *TSF is set 2 ck_apre cycles after the time-stamp event occurs due to synchronization process.*

There is no delay in the setting of TSOVF. This means that if two time-stamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a time-stamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a time-stamp event occurring at the same moment, the application must not write '0' into TSF bit unless it has already read it to '1'.

Optionally, a tamper event can cause a time-stamp to be recorded. See the description of the TAMPTS control bit in [Section 27.6.14: RTC time-stamp sub second register \(RTC_TSSSR\)](#).

27.3.14 Tamper detection

The RTC_TAMPx input events can be configured either for edge detection, or for level detection with filtering.

The tamper detection can be configured for the following purposes:

- erase the RTC backup registers
- generate an interrupt, capable to wakeup from Stop and Standby modes

RTC backup registers

The backup registers (RTC_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs (see [Section 27.6.19: RTC backup registers \(RTC_BKPxR\)](#) and [Tamper detection initialization on page 736](#), or when the readout protection of the flash is changed from level 1 to level 0).

Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the RTC_TAFCR register.

Each RTC_TAMPx tamper detection input is associated with a flag TAMPxF in the RTC_ISR register.

The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:

- 3 ck_apre cycles when TAMPFLT differs from 0x0 (Level detection with filtering)
- 3 ck_apre cycles when TAMPTS=1 (Timestamp on tamper event)
- No latency when TAMPFLT=0x0 (Edge detection) and TAMPTS=0

A new tamper occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

By setting the TAMPIE bit in the RTC_TAFCR register, an interrupt is generated when a tamper detection event occurs. .

Timestamp on tamper event

With TAMPTS set to '1', any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit are set in RTC_ISR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set at the same time that TSF or TSOVF is set.

Edge detection on tamper inputs

If the TAMPFLT bits are "00", the RTC_TAMPx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the RTC_TAMPx inputs are deactivated when edge detection is selected.

- Caution:** To avoid losing tamper detection events, the signal used for edge detection is logically ANDed with the corresponding TAMPxE bit in order to detect a tamper detection event in case it occurs before the RTC_TAMPx pin is enabled.
- When TAMPxTRG = 0: if the RTC_TAMPx alternate function is already high before tamper detection is enabled (TAMPxE bit set to 1), a tamper event is detected as soon as the RTC_TAMPx input is enabled, even if there was no rising edge on the RTC_TAMPx input after TAMPxE was set.
 - When TAMPxTRG = 1: if the RTC_TAMPx alternate function is already low before tamper detection is enabled, a tamper event is detected as soon as the RTC_TAMPx input is enabled (even if there was no falling edge on the RTC_TAMPx input after TAMPxE was set).

After a tamper event has been detected and cleared, the RTC_TAMPx alternate function should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (RTC_BKPxR). This prevents the application from writing to the backup registers while the RTC_TAMPx input value still indicates a tamper detection. This is equivalent to a level detection on the RTC_TAMPx alternate function input.

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

Level detection with filtering on RTC_TAMPx inputs

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The RTC_TAMPx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the RTC_TAMPx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

Note: *Refer to the datasheets for the electrical characteristics of the pull-up resistors.*

27.3.15 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the RTC_CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the RTC_CALIB frequency is $f_{\text{RTCCLK}}/64$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The RTC_CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and "PREDIV_S+1" is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the RTC_CALIB frequency is $f_{\text{RTCCLK}}/(256 * (\text{PREDIV_A}+1))$. 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 RTC_CALIB or RTC_ALARM output is selected, the RTC_OUT pin is automatically configured in output alternate function.*

27.3.16 Alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm alternate function output RTC_ALARM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_ISR register.

The polarity of the output is determined by the POL control bit in RTC_CR so that the opposite of the selected flag bit is output when POL is set to 1.

Alarm alternate function output

The RTC_ALARM pin can be configured in output open drain or output push-pull using the control bit ALARMOUTTYPE in the RTC_TAFCR register.

Note: Once the RTC_ALARM output is enabled, it has priority over RTC_CALIB (COE bit is don't care and must be kept cleared).

When the RTC_CALIB or RTC_ALARM output is selected, the RTC_OUT pin is automatically configured in output alternate function.

27.4 RTC low-power modes

Table 133. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Standby mode.

27.5 RTC interrupts

All RTC interrupts are connected to the EXTI controller. Refer to [Section 14.2: Extended interrupts and events controller \(EXTI\)](#).

To enable the RTC Alarm interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC Alarm event in interrupt mode and select the rising edge sensitivity.
2. Configure and enable the RTC_ALARM IRQ channel in the NVIC.
3. Configure the RTC to generate RTC alarms.

To enable the RTC Tamper interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC Tamper event in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC_TAMP_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC tamper event.

To enable the RTC TimeStamp interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC TimeStamp event in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC_TAMP_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC time-stamp event.

To enable the Wakeup timer interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the Wakeup timer even in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC_WKUP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC Wakeup timer event.

Table 134. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit from Sleep mode	Exit from Stop mode	Exit from Standby mode
Alarm A	ALRAF	ALRAIE	yes	yes ⁽¹⁾	yes ⁽¹⁾
Alarm B	ALRBF	ALRBIE	yes	yes ⁽¹⁾	yes ⁽¹⁾
RTC_TS input (timestamp)	TSF	TSIE	yes	yes ⁽¹⁾	yes ⁽¹⁾
RTC_TAMP1 input detection	TAMP1F	TAMPIE	yes	yes ⁽¹⁾	yes ⁽¹⁾
RTC_TAMP2 input detection	TAMP2F	TAMPIE	yes	yes ⁽¹⁾	yes ⁽¹⁾
RTC_TAMP3 input detection	TAMP3F	TAMPIE	yes	yes ⁽¹⁾	yes ⁽¹⁾
Wakeup timer interrupt	WUTF	WUTIE	yes	yes ⁽¹⁾	yes ⁽¹⁾

1. Wakeup from STOP and Standby modes is possible only when the RTC clock source is LSE or LSI.

27.6 RTC registers

Refer to [Section 2.1 on page 42](#) of the reference manual for a list of abbreviations used in register descriptions.

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

27.6.1 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 729](#) and [Reading the calendar on page 730](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 729](#).

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

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

Bits 31-23 Reserved, must be kept at reset value

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bit 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bit 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bit 3:0 **SU[3:0]**: Second units in BCD format

27.6.2 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 729](#) and [Reading the calendar on page 730](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 729](#).

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset: 0x0000 2101 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

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

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

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units

000: forbidden

001: Monday

...

111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

27.6.3 RTC control register (RTC_CR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
								r/w	r/w	r/w	r/w	r/w	r/w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPH HAD	REFCKON	TSEDGE	WUCKSEL[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

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

Bit 23 **COE**: Calibration output enable

This bit enables the RTC_CALIB output

0: Calibration output disabled

1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to RTC_ALARM output

00: Output disabled

01: Alarm A output enabled

10: Alarm B output enabled

11: Wakeup output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of RTC_ALARM output

0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0])

1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]).

Bit 19 **COSEL**: Calibration output selection

When COE=1, this bit selects which signal is output on RTC_CALIB.

0: Calibration output is 512 Hz

1: Calibration output is 1 Hz

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A=127 and PREDIV_S=255). Refer to [Section 27.3.15: Calibration clock output](#)

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change.

- Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.
0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change
- Bit 15 **TSIE**: Time-stamp interrupt enable
0: Time-stamp Interrupt disable
1: Time-stamp Interrupt enable
- Bit 14 **WUTIE**: Wakeup timer interrupt enable
0: Wakeup timer interrupt disabled
1: Wakeup timer interrupt enabled
- Bit 13 **ALRBIE**: *Alarm B interrupt enable*
0: Alarm B Interrupt disable
1: Alarm B Interrupt enable
- Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled
- Bit 11 **TSE**: timestamp enable
0: timestamp disable
1: timestamp enable
- Bit 10 **WUTE**: Wakeup timer enable
0: Wakeup timer disabled
1: Wakeup timer enabled
- Bit 9 **ALRBE**: *Alarm B enable*
0: Alarm B disabled
1: Alarm B enabled
- Bit 8 **ALRAE**: Alarm A enable
0: Alarm A disabled
1: Alarm A enabled
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format
- Bit 5 **BYPHAD**: Bypass the shadow registers
0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.
- Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to '1'.*

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)

0: RTC_REFIN detection disabled

1: RTC_REFIN detection enabled

Note: PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Time-stamp event active edge

0: RTC_TS input rising edge generates a time-stamp event

1: RTC_TS input falling edge generates a time-stamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: Wakeup clock selection

000: RTC/16 clock is selected

001: RTC/8 clock is selected

010: RTC/4 clock is selected

011: RTC/2 clock is selected

10x: ck_spre (usually 1 Hz) clock is selected

11x: ck_spre (usually 1 Hz) clock is selected and 2^{16} is added to the WUT counter value
(see note below)

Note: Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC_ISR/INITF = 1).

WUT = Wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ISR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

This register is write protected. The write access procedure is described in [RTC register write protection on page 729](#).

27.6.4 RTC initialization and status register (RTC_ISR)

This register is write protected (except for RTC_ISR[13:8] bits). The write access procedure is described in [RTC register write protection on page 729](#).

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to '0'

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRB WF	ALRAWF
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	r	r	r	r

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

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to '1' when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to '0'. Refer to [Re-calibration on-the-fly](#).

Bit 15 **TAMP3F**: RTC_TAMP3 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP3 input.

It is cleared by software writing 0

Bit 14 **TAMP2F**: RTC_TAMP2 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP2 input.

It is cleared by software writing 0

Bit 13 **TAMP1F**: RTC_TAMP1 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP1 input.

It is cleared by software writing 0

Bit 12 **TSOVF**: Time-stamp overflow flag

This flag is set by hardware when a time-stamp event occurs while TSF is already set.

This flag is cleared by software by writing 0. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a time-stamp event occurs immediately before the TSF bit is cleared.

Bit 11 **TSF**: Time-stamp flag

This flag is set by hardware when a time-stamp event occurs.

This flag is cleared by software by writing 0.

Bit 10 **WUTF**: Wakeup timer flag

This flag is set by hardware when the wakeup auto-reload counter reaches 0.

This flag is cleared by software by writing 0.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

- Bit 9 **ALRBF**: Alarm B flag
This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm B register (RTC_ALRMBR).
This flag is cleared by software by writing 0.
- Bit 8 **ALRAF**: Alarm A flag
This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm A register (RTC_ALRMAR).
This flag is cleared by software by writing 0.
- Bit 7 **INIT**: Initialization mode
0: Free running mode
1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.
- Bit 6 **INITF**: Initialization flag
When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.
0: Calendar registers update is not allowed
1: Calendar registers update is allowed
- Bit 5 **RSF**: Registers synchronization flag
This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow register mode (BYPHAD=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 (Backup domain reset state).
0: Calendar has not been initialized
1: Calendar has been initialized
- Bit 3 **SHPF**: Shift operation pending
0: No shift operation is pending
1: A shift operation is pending
This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware when the wakeup timer values can be changed, after the WUTE bit has been set to 0 in RTC_CR.

0: Wakeup timer configuration update not allowed

1: Wakeup timer configuration update allowed

Bit 1 **ALRBWF**: Alarm B write flag

This bit is set by hardware when Alarm B values can be changed, after the ALRBE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm B update not allowed

1: Alarm B update allowed

Bit 0 **ALRAWF**: Alarm A write flag

This bit is set by hardware when Alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm A update not allowed

1: Alarm A update allowed

Note: *The bits ALRAF, ALRBF, WUTF and TSF are cleared 2 APB clock cycles after programming them to 0.*

27.6.5 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 729](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 729](#).

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

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

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

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:

$ck_apre\ frequency = RTCCLK\ frequency / (PREDIV_A + 1)$

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:

$ck_spre\ frequency = ck_apre\ frequency / (PREDIV_S + 1)$

27.6.6 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ISR.

This register is write protected. The write access procedure is described in [RTC register write protection on page 729](#).

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

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

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

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs (WUT+1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

27.6.7 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 729](#).

Address offset: 0x1C

Backup 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.

27.6.8 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 729](#).

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask

0: Alarm B set if the date and day match

1: Date and day don't care in Alarm B comparison

Bit 30 **WDSEL**: Week day selection

0: DU[3:0] represents the date units

1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask

0: Alarm B set if the hours match

1: Hours don't care in Alarm B comparison

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask

0: Alarm B set if the minutes match

1: Minutes don't care in Alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask

0: Alarm B set if the seconds match

1: Seconds don't care in Alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

27.6.9 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 7:0 **KEY**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to [RTC register write protection](#) for a description of how to unlock RTC register write protection.

27.6.10 RTC sub second register (RTC_SSR)

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

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

Bits31:16 Reserved, must be kept at reset value

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

Note: SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.



27.6.11 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 729](#).

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_ISR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 **Reserved**, must be kept at reset value

Bits 14:0 **SUBFS**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_ISR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF=1 to be sure that the shadow registers have been updated with the shifted time.

27.6.12 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

Address offset: 0x30

Backup 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.

27.6.13 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

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

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

Bits 15:13 **WDU[1:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bit 3:0 **DU[3:0]**: Date units in BCD format

27.6.14 RTC time-stamp sub second register (RTC_TSSSR)

The content of this register is valid only when RTC_ISR/TSF is set. It is cleared when the RTC_ISR/TSF bit is reset.

Address offset: 0x38
Backup domain reset value: 0x0000 0000
System reset: not affected

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

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

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

27.6.15 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 729](#).

Address offset: 0x3C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:16 Reserved, must be kept at reset value

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. If the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: $(512 * CALP) - CALM$.

Refer to [Section 27.3.12: RTC smooth digital calibration](#).

Bit 14 **CALW8**: Use an 8-second calibration cycle period

When CALW8 is set to '1', the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at "00" when CALW8='1'. Refer to [Section 27.3.12: RTC smooth digital calibration](#).

Bit 13 **CALW16**: Use a 16-second calibration cycle period

When CALW16 is set to '1', the 16-second calibration cycle period is selected. This bit must not be set to '1' if CALW8=1.

Note: CALM[0] is stuck at '0' when CALW16='1'. Refer to [Section 27.3.12: RTC smooth digital calibration](#).

Bits 12:9 Reserved, must be kept at reset value

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See [Section 27.3.12: RTC smooth digital calibration on page 733](#).

27.6.16 RTC tamper and alternate function configuration register (RTC_TAFCR)

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC15 MODE	PC15 VALUE	PC14 MODE	PC14 VALUE	PC13 MODE	PC13 VALUE	Res.	Res.
								rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMPP UDIS	TAMPPRCH [1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPT S			TAMP2 TRG	TAMP2 E	TAMPIE	TAMP1 TRG	TAMP1 E
rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw

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

Bit 23 **PC15MODE**: PC15 mode

0: PC15 is controlled by the GPIO configuration registers. Consequently PC15 is floating in Standby mode.

1: PC15 is forced to push-pull output if LSE is disabled.

Bit 22 **PC15VALUE**: PC15 value

If the LSE is disabled and PC15MODE = 1, PC15VALUE configures the PC15 output data.

Bit 21 **PC14MODE**: PC14 mode

0: PC14 is controlled by the GPIO configuration registers. Consequently PC14 is floating in Standby mode.

1: PC14 is forced to push-pull output if LSE is disabled.

Bit 20 **PC14VALUE**: PC14 value

If the LSE is disabled and PC14MODE = 1, PC14VALUE configures the PC14 output data.

Bit 19 **PC13MODE**: PC13 mode

0: PC13 is controlled by the GPIO configuration registers. Consequently PC13 is floating in Standby mode.

1: PC13 is forced to push-pull output if all RTC alternate functions are disabled.

Bit 18 **PC13VALUE**: RTC_ALARM output type/PC13 value

If PC13 is used to output RTC_ALARM, PC13VALUE configures the output configuration:

0: RTC_ALARM is an open-drain output

1: RTC_ALARM is a push-pull output

If all RTC alternate functions are disabled and PC13MODE = 1, PC13VALUE configures the PC13 output data.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **TAMPPUDIS**: RTC_TAMPx pull-up disable

This bit determines if each of the RTC_TAMPx pins are pre-charged before each sample.

0: Precharge RTC_TAMPx pins before sampling (enable internal pull-up)

1: Disable precharge of RTC_TAMPx pins.

Bits 14:13 **TAMPPRCH[1:0]**: RTC_TAMPx precharge duration

These bits determine the duration of time during which the pull-up is activated before each sample. TAMPPRCH is valid for each of the RTC_TAMPx inputs.

- 0x0: 1 RTCCLK cycle
- 0x1: 2 RTCCLK cycles
- 0x2: 4 RTCCLK cycles
- 0x3: 8 RTCCLK cycles

Bits 12:11 **TAMPFLT[1:0]**: RTC_TAMPx filter count

These bits determine the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a Tamper event. TAMPFLT is valid for each of the RTC_TAMPx inputs.

- 0x0: Tamper event is activated on edge of RTC_TAMPx input transitions to the active level (no internal pull-up on RTC_TAMPx input).
- 0x1: Tamper event is activated after 2 consecutive samples at the active level.
- 0x2: Tamper event is activated after 4 consecutive samples at the active level.
- 0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 10:8 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the RTC_TAMPx inputs are sampled.

- 0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)
- 0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)
- 0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)
- 0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)
- 0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)
- 0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)
- 0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)
- 0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Bit 7 **TAMPTS**: Activate timestamp on tamper detection event

- 0: Tamper detection event does not cause a timestamp to be saved
- 1: Save timestamp on tamper detection event

TAMPTS is valid even if TSE=0 in the RTC_CR register.

Bit 6 **TAMP3TRG**: Active level for RTC_TAMP3 input

- if TAMPFLT != 00:
 - 0: RTC_TAMP3 input staying low triggers a tamper detection event.
 - 1: RTC_TAMP3 input staying high triggers a tamper detection event.
- if TAMPFLT = 00:
 - 0: RTC_TAMP3 input rising edge triggers a tamper detection event.
 - 1: RTC_TAMP3 input falling edge triggers a tamper detection event.

Bit 5 **TAMP3E**: RTC_TAMP3 detection enable

- 0: RTC_TAMP3 input detection disabled
- 1: RTC_TAMP3 input detection enabled

Bit 4 **TAMP2TRG**: Active level for RTC_TAMP2 input

- if TAMPFLT != 00:
 - 0: RTC_TAMP2 input staying low triggers a tamper detection event.
 - 1: RTC_TAMP2 input staying high triggers a tamper detection event.
- if TAMPFLT = 00:
 - 0: RTC_TAMP2 input rising edge triggers a tamper detection event.
 - 1: RTC_TAMP2 input falling edge triggers a tamper detection event.

- Bit 3 **TAMP2E**: RTC_TAMP2 input detection enable
0: RTC_TAMP2 detection disabled
1: RTC_TAMP2 detection enabled
- Bit 2 **TAMPIE**: Tamper interrupt enable
0: Tamper interrupt disabled
1: Tamper interrupt enabled.
- Bit 1 **TAMP1TRG**: Active level for RTC_TAMP1 input
If TAMPFLT != 00
0: RTC_TAMP1 input staying low triggers a tamper detection event.
1: RTC_TAMP1 input staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: RTC_TAMP1 input rising edge triggers a tamper detection event.
1: RTC_TAMP1 input falling edge triggers a tamper detection event.
- Bit 0 **TAMP1E**: RTC_TAMP1 input detection enable
0: RTC_TAMP1 detection disabled
1: RTC_TAMP1 detection enabled

Caution: When TAMPFLT = 0, TAMP1E must be reset when TAMP1TRG is changed to avoid spuriously setting TAMP1F.

27.6.17 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 729](#)

Address offset: 0x44

Backup 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.
				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.

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.

27.6.18 RTC alarm B sub second register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).

Address offset: 0x48

Backup 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.
				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.	SS[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	w	r/w	r/w

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.

27.6.19 RTC backup registers (RTC_BKPxR)

Address offset: 0x50 to 0x8C

Backup 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.
This register is reset on a tamper detection event, as long as TAMPxF=1. or when the Flash readout protection is disabled.

27.6.20 RTC register map

Table 135. 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.	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	0	1	0	0	0	0	1			0	0	0	0	0	0
0x08	RTC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COE	OSE L [1:0]	POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPHAD	REFCKON	TSEDGE	WUCKS EL[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	
0x0C	RTC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INTS	SHPF	WUTF	ALRWF	ALRAWF		
	Reset value															0		0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
0x10	RTC_PRER	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	1	1	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.	WUT[15:0]																	
	Reset value																1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
0x1C	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]	DU[3:0]			MSK3	PM	HT [1:0]	HU[3:0]			MSK2	MNT[2:0]			MNU[3:0]			MSK1	ST[2:0]			SU[3:0]									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]	DU[3:0]			MSK3	PM	HT [1:0]	HU[3:0]			MSK2	MNT[2:0]			MNU[3:0]			MSK2	ST[2:0]			SU[3:0]									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
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.	KEY									
	Reset value																								0	0	0	0	0	0	0	0	0	
0x28	RTC_SSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]																	
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x2C	RTC_SHIFTR	ADD1S	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	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.	WDU[1:0]			MT	MU[3:0]			Res.	Res.	DT [1:0]			DU[3:0]					
	Reset value																0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	
0x38	RTC_TSSSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]																	
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 135. RTC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
0x3C	RTC_CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]													
	Reset value																	0	0	0					0	0	0	0	0	0	0	0	0					
0x40	RTC_TAFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC15MODE	PC15MODE	PC14VALUE	PC14MODE	PC13VALUE	PC13VALUE	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		TAMPTS		TAMP3-TRG	TAMP3E	TAMP2-TRG	TAMP2E	TAMP1E	TAMP1TRG	TAMP1E					
	Reset value									0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x44	RTC_ALRMASSR	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]																			
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x48	RTC_ALRMBSSR	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]																			
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x50 to 0x8C	RTC_BKP0R	BKP[31:0]																																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
	to RTC_BKP15R	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						

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

28 Inter-integrated circuit (I2C) interface

28.1 Introduction

The I²C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I²C bus. It provides multimaster capability, and controls all I²C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus (power management bus) compatible.

DMA can be used to reduce CPU overload.

28.2 I2C main features

- I²C bus specification rev03 compatibility:
 - Slave and master modes
 - Multimaster capability
 - Standard-mode (up to 100 kHz)
 - Fast-mode (up to 400 kHz)
 - Fast-mode Plus (up to 1 MHz)
 - 7-bit and 10-bit addressing mode
 - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
 - All 7-bit addresses acknowledge mode
 - General call
 - Programmable setup and hold times
 - Easy to use event management
 - Optional clock stretching
 - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters

The following additional features are also available depending on the product implementation (see [Section 28.3: I2C implementation](#)):

- SMBus specification rev 2.0 compatibility:
 - Hardware PEC (Packet Error Checking) generation and verification with ACK control
 - Command and data acknowledge control
 - Address resolution protocol (ARP) support
 - Host and Device support
 - SMBus alert
 - Timeouts and idle condition detection
- PMBus rev 1.1 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the PCLK reprogramming
- Wakeup from Stop mode on address match.

28.3 I2C implementation

This manual describes the full set of features implemented in I2C1, I2C3 and I2C3., I2C2, I2C3 and I2C4./I2C3.

Table 136. STM32F302xx I2C implementation

I2C features ⁽¹⁾	I2C1	I2C2	I2C3 ⁽²⁾
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 20mA output drive I/Os (up to 1 Mbit/s)	X	X	X
Independent clock	X	X	X
SMBus	X	X	X
Wakeup from Stop mode	X	X	X

1. X = supported.

2. I2C3 is not available in STM32F302xB/C devices.

28.4 I2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I²C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I²C bus.

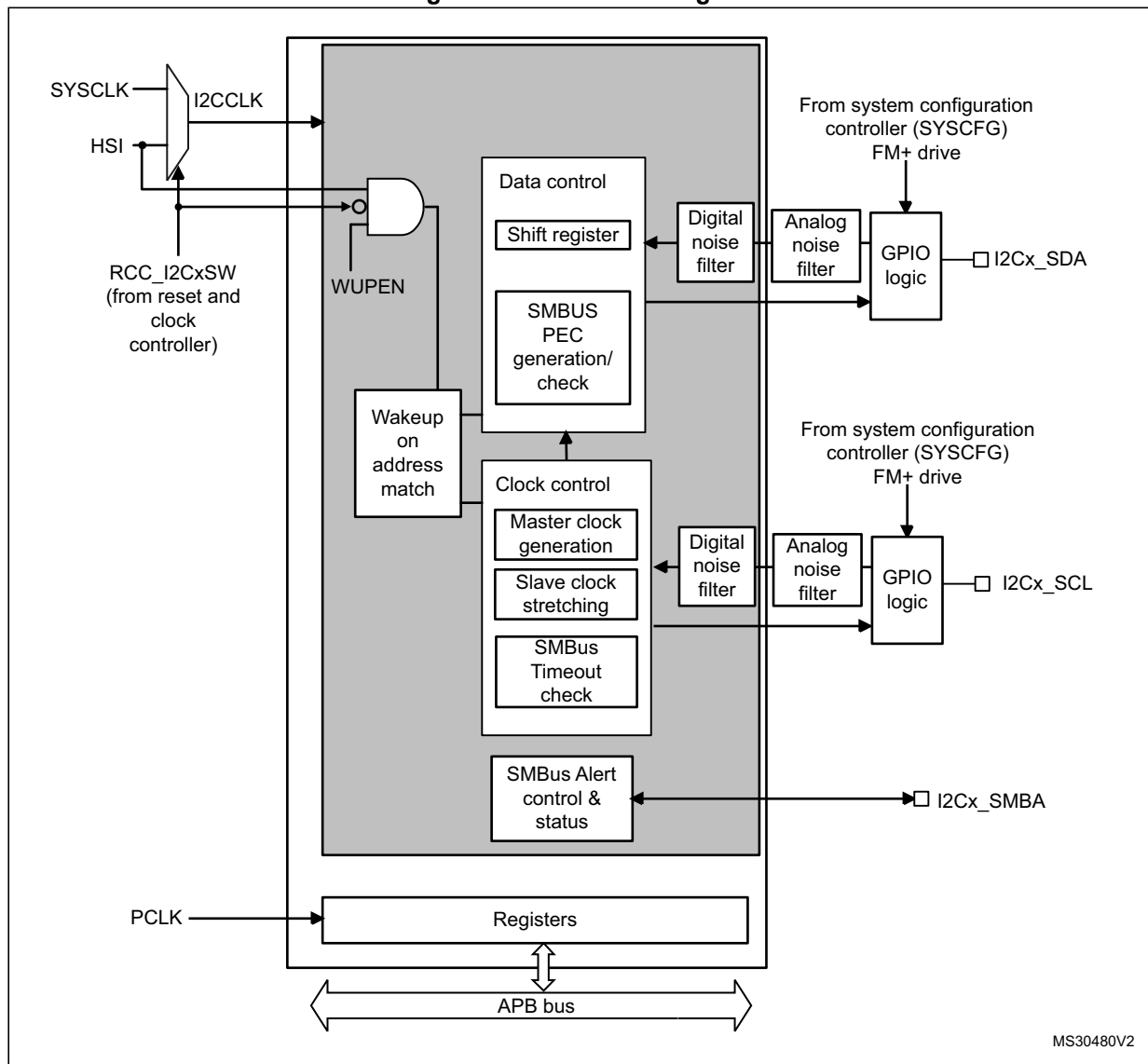
This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

28.4.1 I2C block diagram

The block diagram of the I2C interface is shown in [Figure 283](#).

Figure 283. I2C block diagram



The I2C is clocked by an independent clock source which allows to the I2C to operate independently from the PCLK frequency.

This independent clock source can be selected for either of the following two clock sources:

- HSI: high speed internal oscillator (default value)
- SYSCLK: system clock

Refer to [Section 10: Reset and clock control \(RCC\)](#) for more details.

I2C I/Os support 20 mA output current drive for Fast-mode Plus operation. This is enabled by setting the driving capability control bits for SCL and SDA in [Section 12.1.1: SYSCFG configuration register 1 \(SYSCFG_CFGR1\)](#)

28.4.2 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period t_{I2CCLK} must respect the following conditions:

$$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4 \text{ and } t_{I2CCLK} < t_{HIGH}$$

with:

t_{LOW} : SCL low time and t_{HIGH} : SCL high time

$t_{filters}$: when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is $DNF \times t_{I2CCLK}$.

The PCLK clock period t_{PCLK} must respect the following condition:

$$t_{PCLK} < 4/3 t_{SCL}$$

with t_{SCL} : SCL period

Caution: When the I2C kernel is clocked by PCLK. PCLK must respect the conditions for t_{I2CCLK} .

28.4.3 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

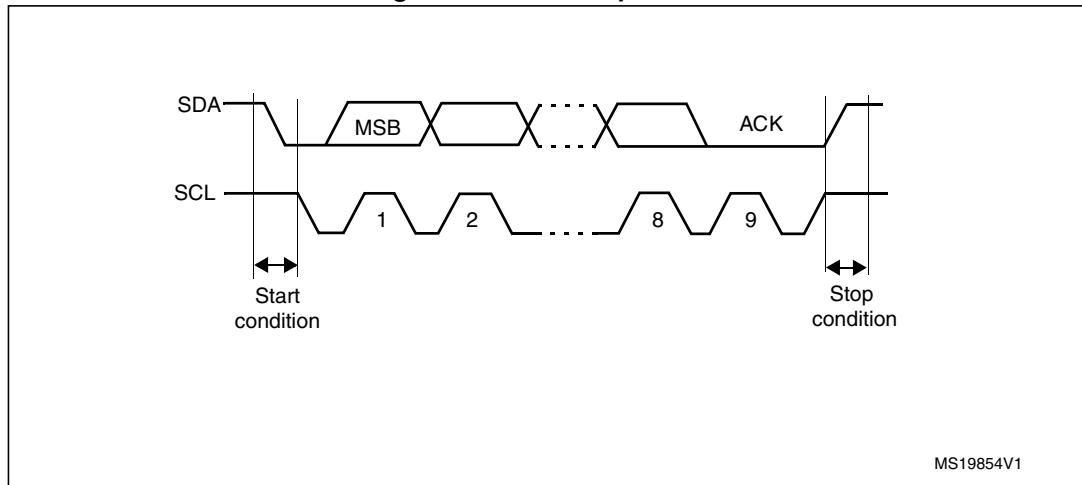
Communication flow

In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

Figure 284. I²C bus protocol

Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

28.4.4 I2C initialization

Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller (refer to [Section 10: Reset and clock control \(RCC\)](#)).

Then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

When the I2C is disabled (PE=0), the I²C performs a software reset. Refer to [Section 28.4.5: Software reset](#) for more details.

Noise filters

Before you enable the I2C peripheral by setting the PE bit in I2C_CR1 register, you must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I²C specification which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. You can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2CCLK periods. This allows to suppress spikes with a programmable length of 1 to 15 I2CCLK periods.

Table 137. Comparison of analog vs. digital filters

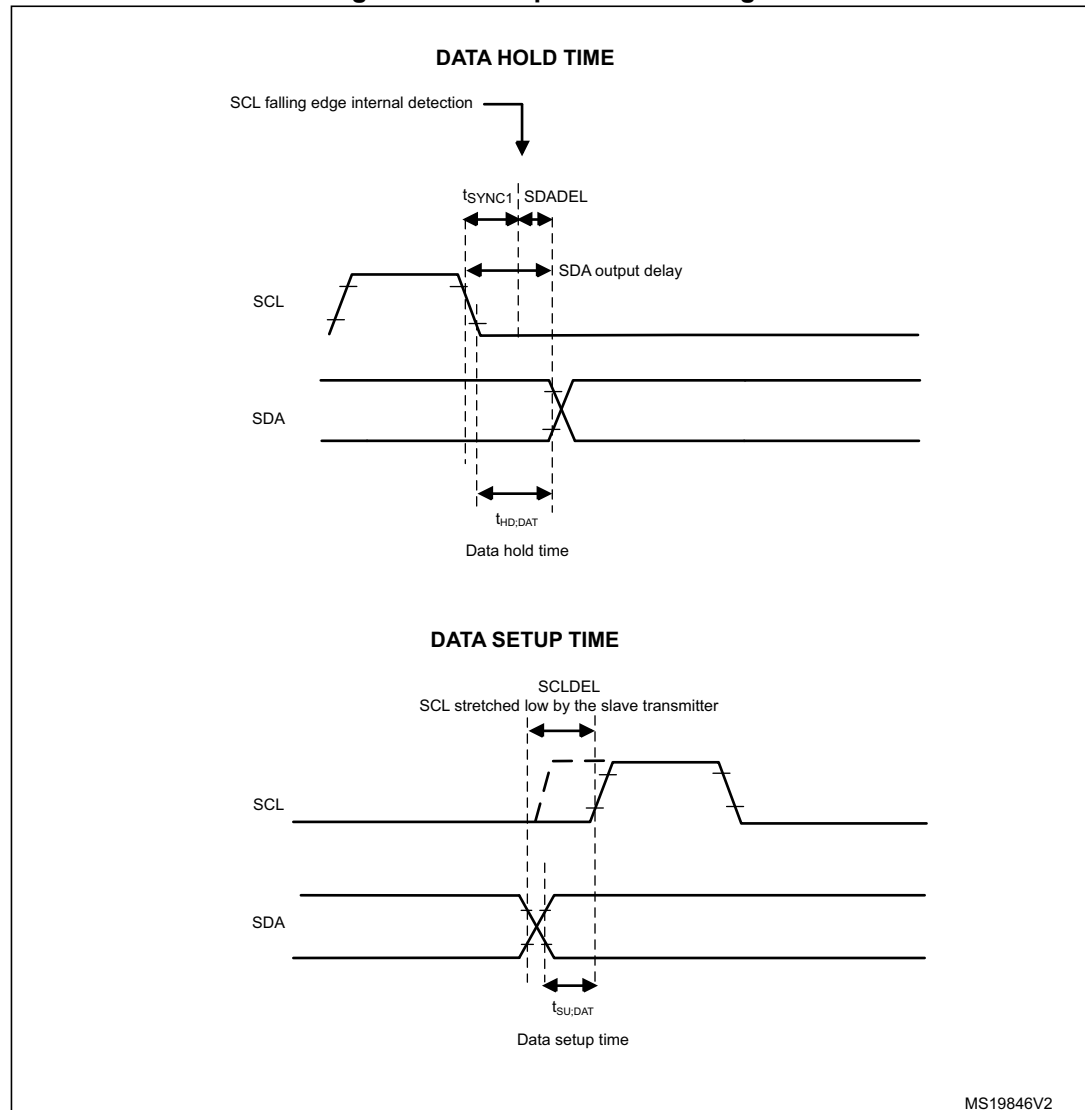
	Analog filter	Digital filter
Pulse width of suppressed spikes	≥ 50 ns	Programmable length from 1 to 15 I2C peripheral clocks
Benefits	Available in Stop mode	<ul style="list-style-type: none"> – Programmable length: extra filtering capability vs. standard requirements – Stable length
Drawbacks	Variation vs. temperature, voltage, process	Wakeup from Stop mode on address match is not available when digital filter is enabled

Caution: Changing the filter configuration is not allowed when the I2C is enabled.

I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

Figure 285. Setup and hold timings



- When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is $t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$.
 t_{SDADEL} impacts the hold time $t_{HD,DAT}$.

The total SDA output delay is:

$$t_{SYNC1} + \{[SDADEL \times (PRESC+1) + 1] \times t_{I2CCLK}\}$$

t_{SYNC1} duration depends on these parameters:

- SCL falling slope
- When enabled, input delay brought by the analog filter: $t_{\text{AF(min)}} < t_{\text{AF}} < t_{\text{AF(max)}} \text{ ns}$.
- When enabled, input delay brought by the digital filter: $t_{\text{DNF}} = \text{DNF} \times t_{\text{I2CCLK}}$
- Delay due to SCL synchronization to I2CCLK clock (2 to 3 I2CCLK periods)

In order to bridge the undefined region of the SCL falling edge, you must program SDADEL in such a way that:

$$\{t_{\text{f(max)}} + t_{\text{HD;DAT(min)}} - t_{\text{AF(min)}} - [(DNF+3) \times t_{\text{I2CCLK}}]\} / \{(PRESC+1) \times t_{\text{I2CCLK}}\} \leq \text{SDADEL}$$

$$\text{SDADEL} \leq \{t_{\text{HD;DAT(max)}} - t_{\text{AF(max)}} - [(DNF+4) \times t_{\text{I2CCLK}}]\} / \{(PRESC+1) \times t_{\text{I2CCLK}}\}$$

Note: $t_{\text{AF(min)}} / t_{\text{AF(max)}}$ are part of the equation only when the analog filter is enabled. Refer to device datasheet for t_{AF} values.

The maximum $t_{\text{HD;DAT}}$ could be 3.45 μs , 0.9 μs and 0.45 μs for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of $t_{\text{VD;DAT}}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

$$\text{SDADEL} \leq \{t_{\text{VD;DAT(max)}} - t_{\text{r(max)}} - 260 \text{ ns} - [(DNF+4) \times t_{\text{I2CCLK}}]\} / \{(PRESC+1) \times t_{\text{I2CCLK}}\}.$$

Note: This condition can be violated when $\text{NOSTRETCH}=0$, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.

Refer to [Table 138: I2C-SMBUS specification data setup and hold times](#) for t_{f} , t_{r} , $t_{\text{HD;DAT}}$ and $t_{\text{VD;DAT}}$ standard values.

- After sending SDA output, SCL line is kept at low level during the setup time. This setup time is $t_{\text{SCLDEL}} = (\text{SCLDEL}+1) \times t_{\text{PRESC}}$ where $t_{\text{PRESC}} = (\text{PRESC}+1) \times t_{\text{I2CCLK}}$. t_{SCLDEL} impacts the setup time $t_{\text{SU;DAT}}$.

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), you must program SCLDEL in such a way that:

$$\{[t_{\text{r(max)}} + t_{\text{SU;DAT(min)}}] / [(\text{PRESC}+1) \times t_{\text{I2CCLK}}]\} - 1 \leq \text{SCLDEL}$$

Refer to [Table 138: I2C-SMBUS specification data setup and hold times](#) for t_{r} and $t_{\text{SU;DAT}}$ standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.

Table 138. I²C-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	-	μs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100		50		250		ns
t_r	Rise time of both SDA and SCL signals	-	1000		300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300		300	-	120	-	300	

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is $t_{SCLL} = (SCLL+1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$. t_{SCLL} impacts the SCL low time t_{LOW} .
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is $t_{SCLH} = (SCLH+1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$. t_{SCLH} impacts the SCL high time t_{HIGH} .

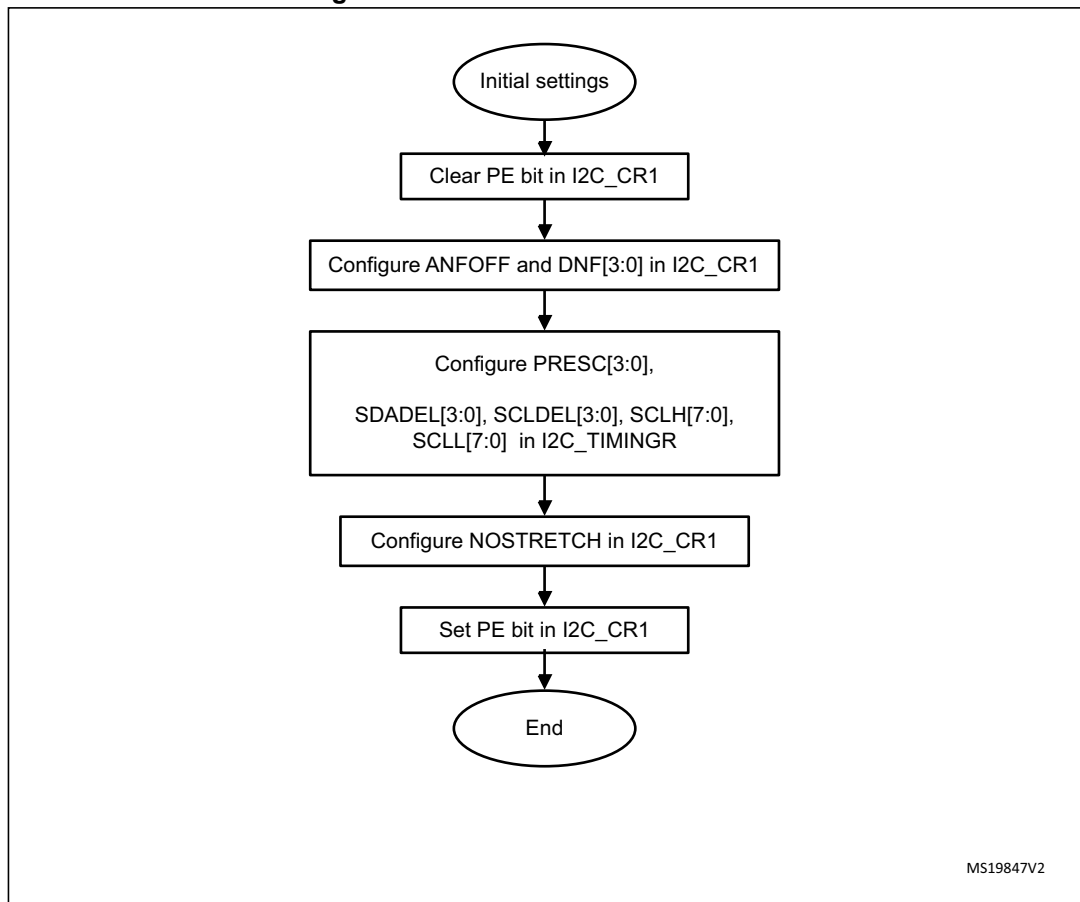
Refer to section : [I2C master initialization](#) for more details.

Caution: Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to : [I2C slave initialization](#) for more details.

Caution: Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

Figure 286. I2C initialization flowchart



28.4.5 Software reset

A software reset can be performed by clearing the PE bit in the I2C_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C_CR2 register: START, STOP, NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C_CR2 register: PECBYTE
2. I2C_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least 3 APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence: - Write PE=0 - Check PE=0 - Write PE=1

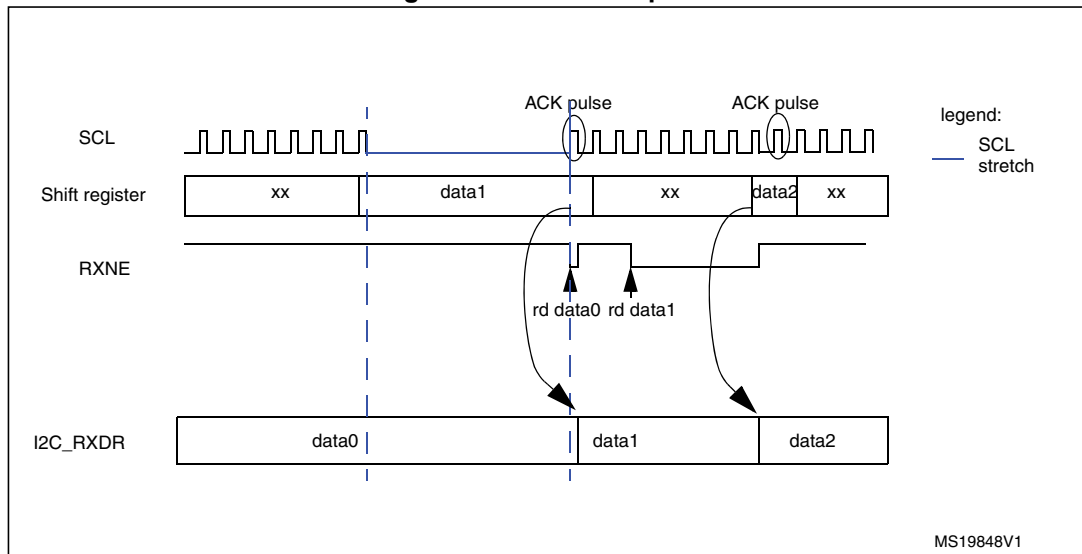
28.4.6 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE=0). If RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

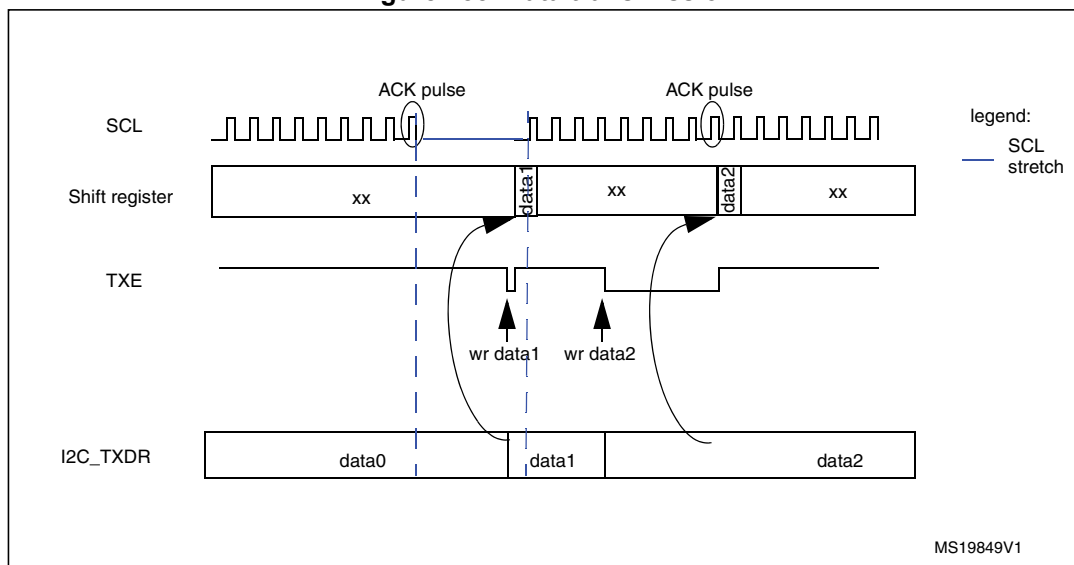
Figure 287. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE=1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the 9th SCL pulse.

Figure 288. Data transmission



Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the SBC (Slave Byte Control) bit in the I2C_CR2 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, TCR flag is set when the number of bytes programmed in NBYTES has been transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.

When RELOAD=0 in master mode, the counter can be used in 2 modes:

- **Automatic end mode** (AUTOEND = '1' in the I2C_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred.
- **Software end mode** (AUTOEND = '0' in the I2C_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C_CR2 register. This mode must be used when the master wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 139. I2C configuration table

Function	SBC bit	RELOAD bit	AUTOEND bit
Master Tx/Rx NBYTES + STOP	x	0	1
Master Tx/Rx + NBYTES + RESTART	x	0	0
Slave Tx/Rx all received bytes ACKed	0	x	x
Slave Rx with ACK control	1	1	x

28.4.7 I2C slave mode

I2C slave initialization

In order to work in slave mode, you must enable at least one slave address. Two registers I2C_OAR1 and I2C_OAR2 are available in order to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.
OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.
- If additional slave addresses are required, you can configure the 2nd slave address OA2. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.
These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK=0.
OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.
- The General Call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOSTRETCH=1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled you must read the ADDCODE[6:0] bits in the I2C_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

Slave clock stretching (NOSTRETCH = 0)

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCONF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE=1). This stretch is released when the data is written to the I2C_TXDR register.
- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during $[(SDADEL+SCLDEL+1) \times (PRESC+1) + 1] \times t_{I2CCLK}$.

Slave without clock stretching (NOSTRETCH = 1)

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if you clear the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, you ensure that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C_RXDR register before the 9th SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Slave Byte Control Mode

In order to allow byte ACK control in slave reception mode, Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD=1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. You can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

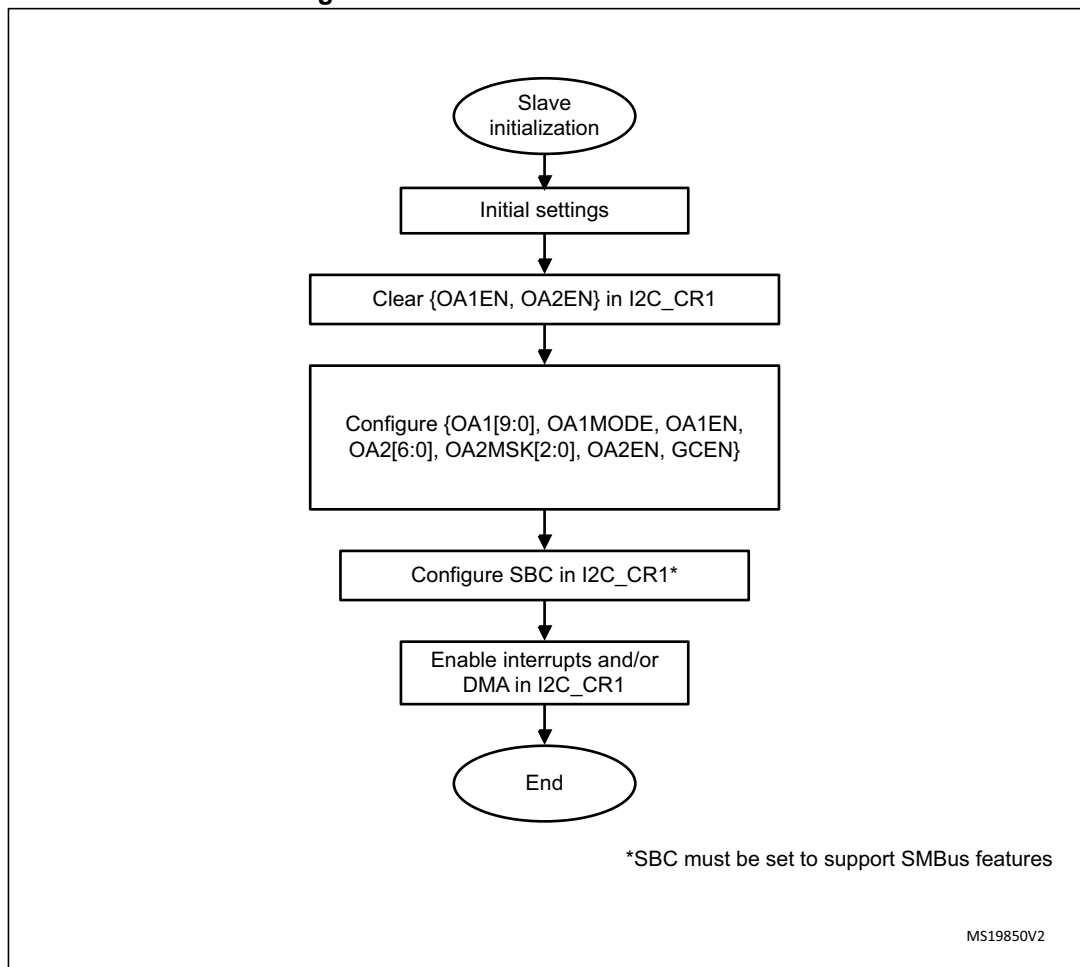
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

Note: The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR=1.

The RELOAD bit value can be changed when ADDR=1, or when TCR=1.

Caution: Slave Byte Control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH=1 is not allowed.

Figure 289. Slave initialization flowchart



Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_CR1 register.

The TXIS bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_CR1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to '0'. In this case, If TXE = 0 when the slave address is received (ADDR=1), you can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the TXE bit in order to program a new data byte.

In Slave Byte Control mode (SBC=1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR=1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

Caution: When NOSTRETCH=1, the SCL clock is not stretched while the ADDR flag is set, so you cannot flush the I2C_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

If STOPF is still set when the first data transmission starts, an underrun error will be generated (the OVR flag is set).

If you need a TXIS event, (Transmit Interrupt or Transmit DMA request), you must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.

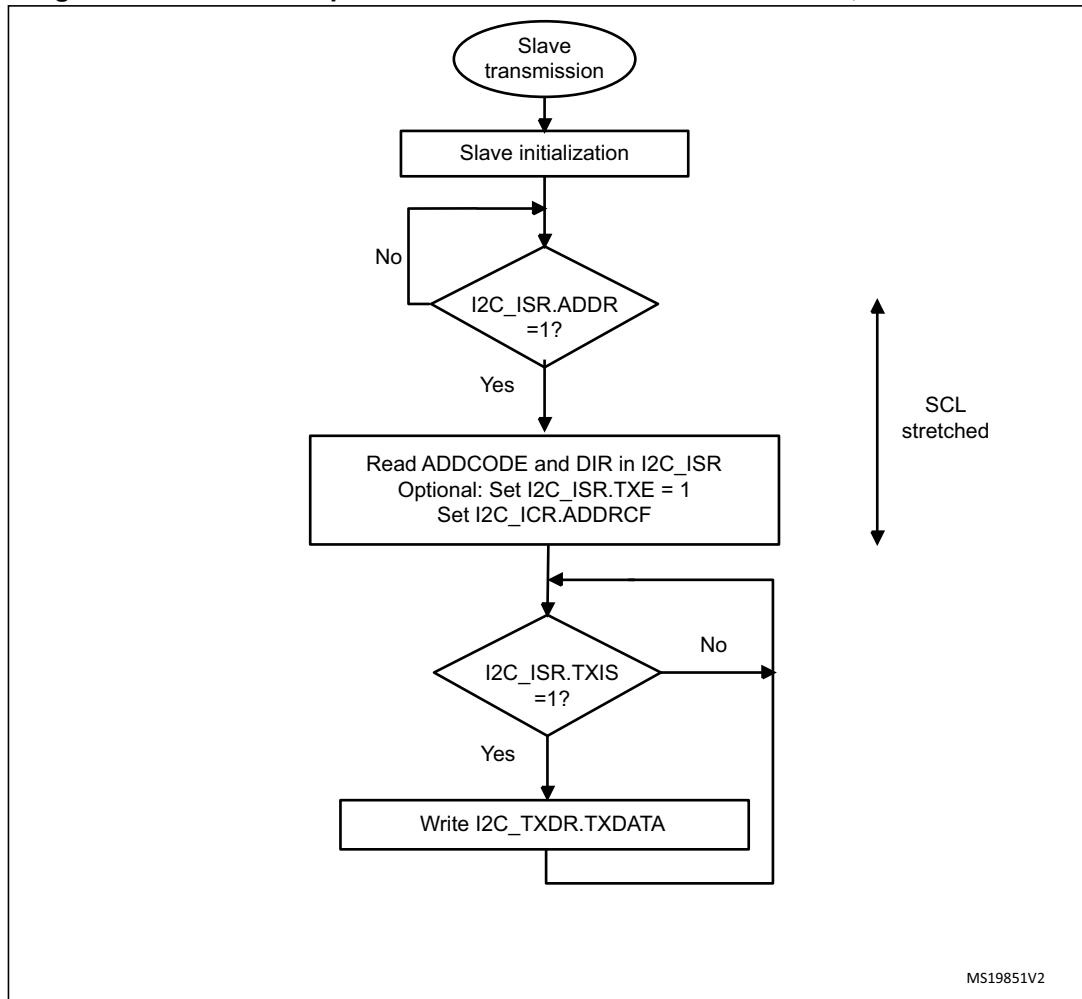
Figure 290. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=0

Figure 291. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=1

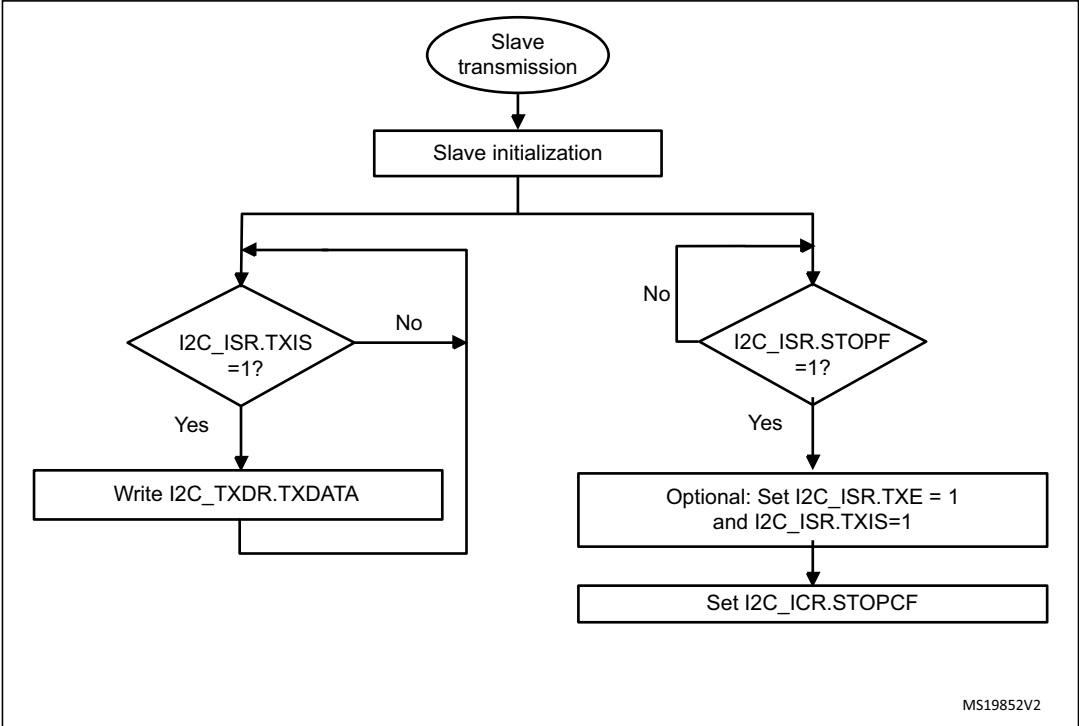
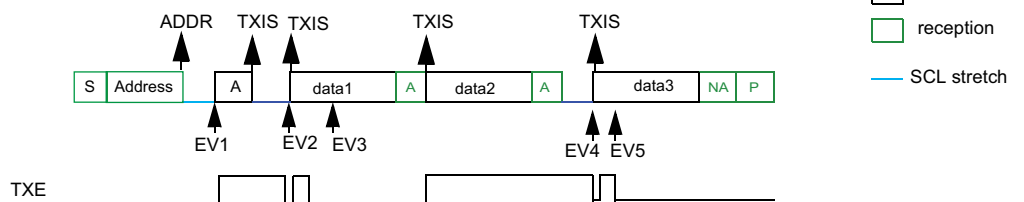


Figure 292. Transfer bus diagrams for I2C slave transmitter

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCONF

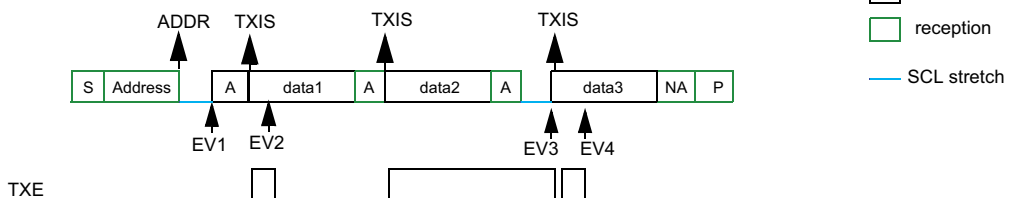
EV2: TXIS ISR: wr data1

EV3: TXIS ISR: wr data2

EV4: TXIS ISR: wr data3

EV5: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



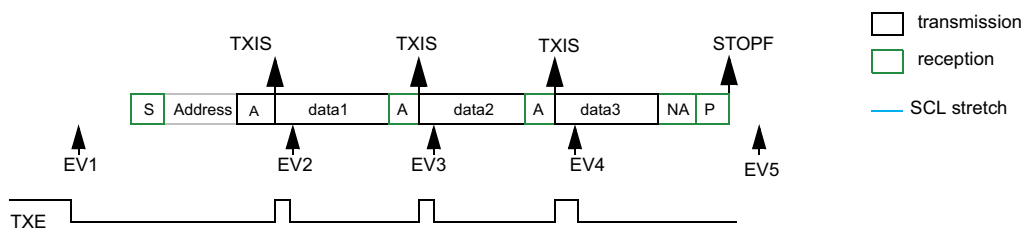
EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCONF

EV2: TXIS ISR: wr data2

EV3: TXIS ISR: wr data3

EV4: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:



EV1: wr data1

EV2: TXIS ISR: wr data2

EV3: TXIS ISR: wr data3

EV4: TXIS ISR: wr data4 (not sent)

EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCONF

MS19853V1

Slave receiver

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

Figure 293. Transfer sequence flowchart for slave receiver with NOSTRETCH=0

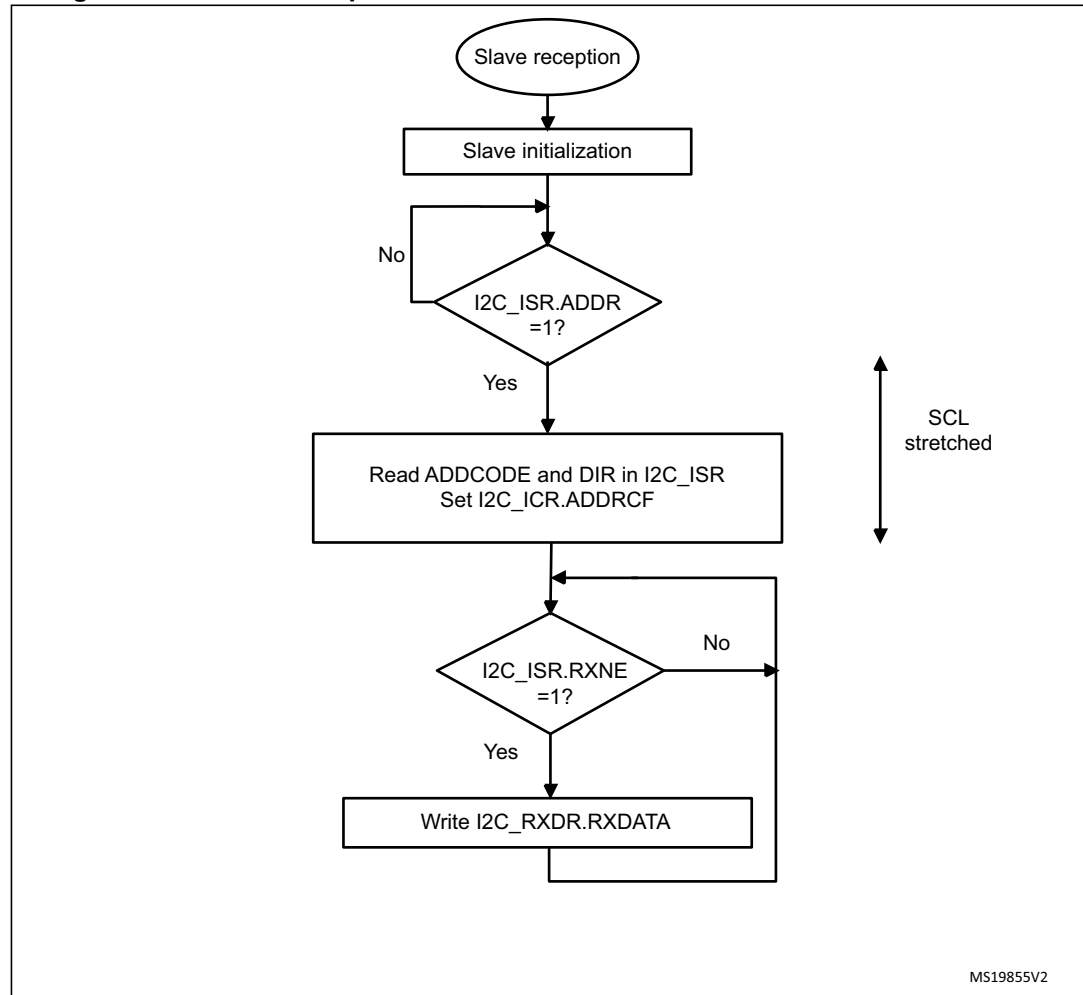


Figure 294. Transfer sequence flowchart for slave receiver with NOSTRETCH=1

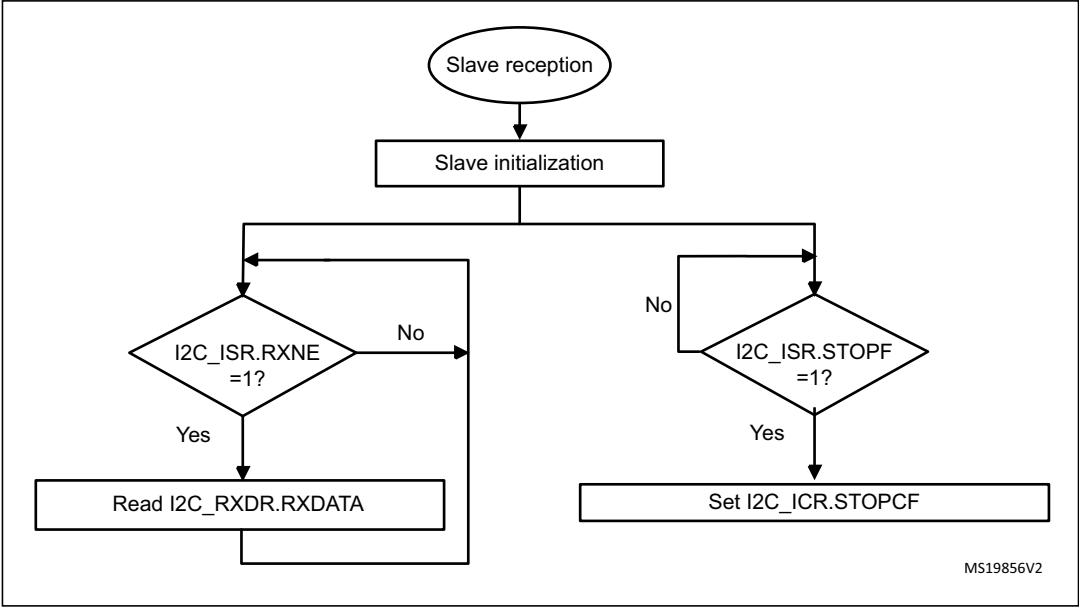
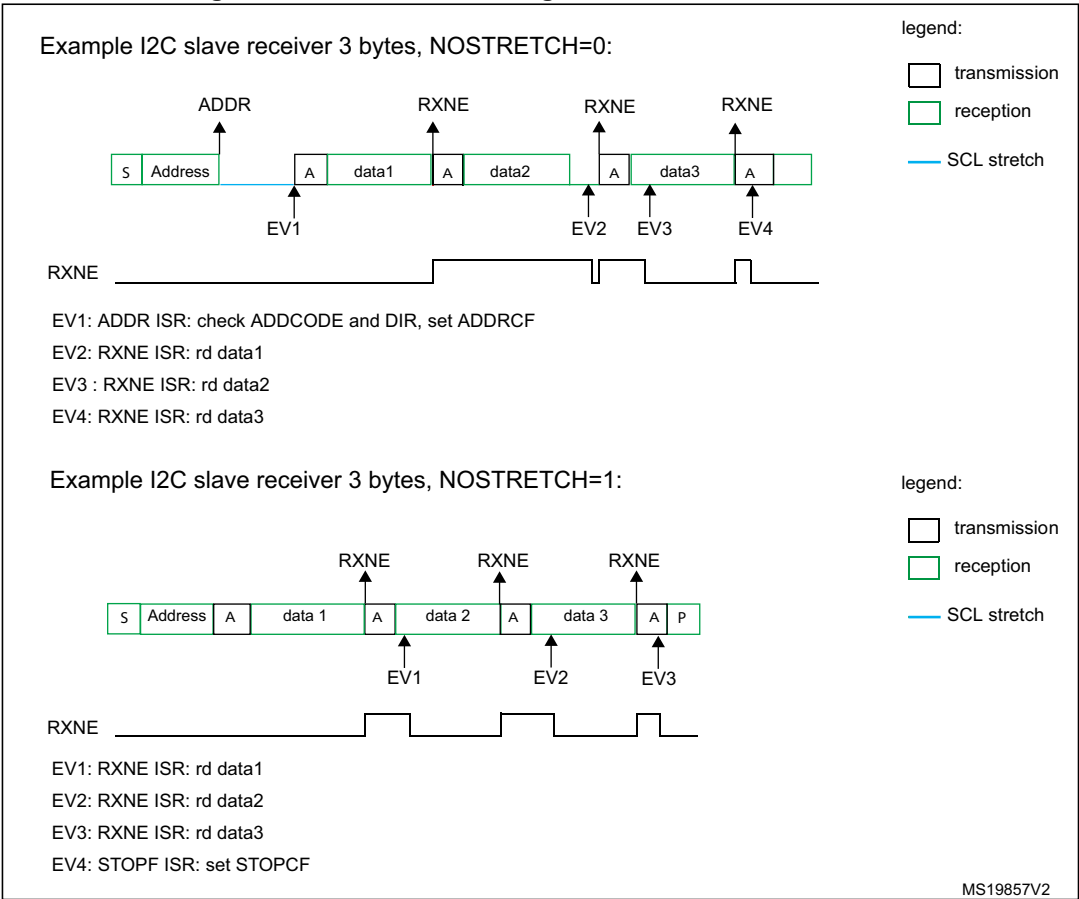


Figure 295. Transfer bus diagrams for I2C slave receiver



28.4.8 I2C master mode

I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2C_TIMINGR register.

The I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter is reached reaches the value programmed in the SCLH[7:0] bits in the I2C_TIMINGR register.

Consequently the master clock period is:

$$t_{\text{SCL}} = t_{\text{SYNC1}} + t_{\text{SYNC2}} + \{[(\text{SCLH}+1) + (\text{SCLL}+1)] \times (\text{PRESC}+1) \times t_{\text{I2CCLK}}\}$$

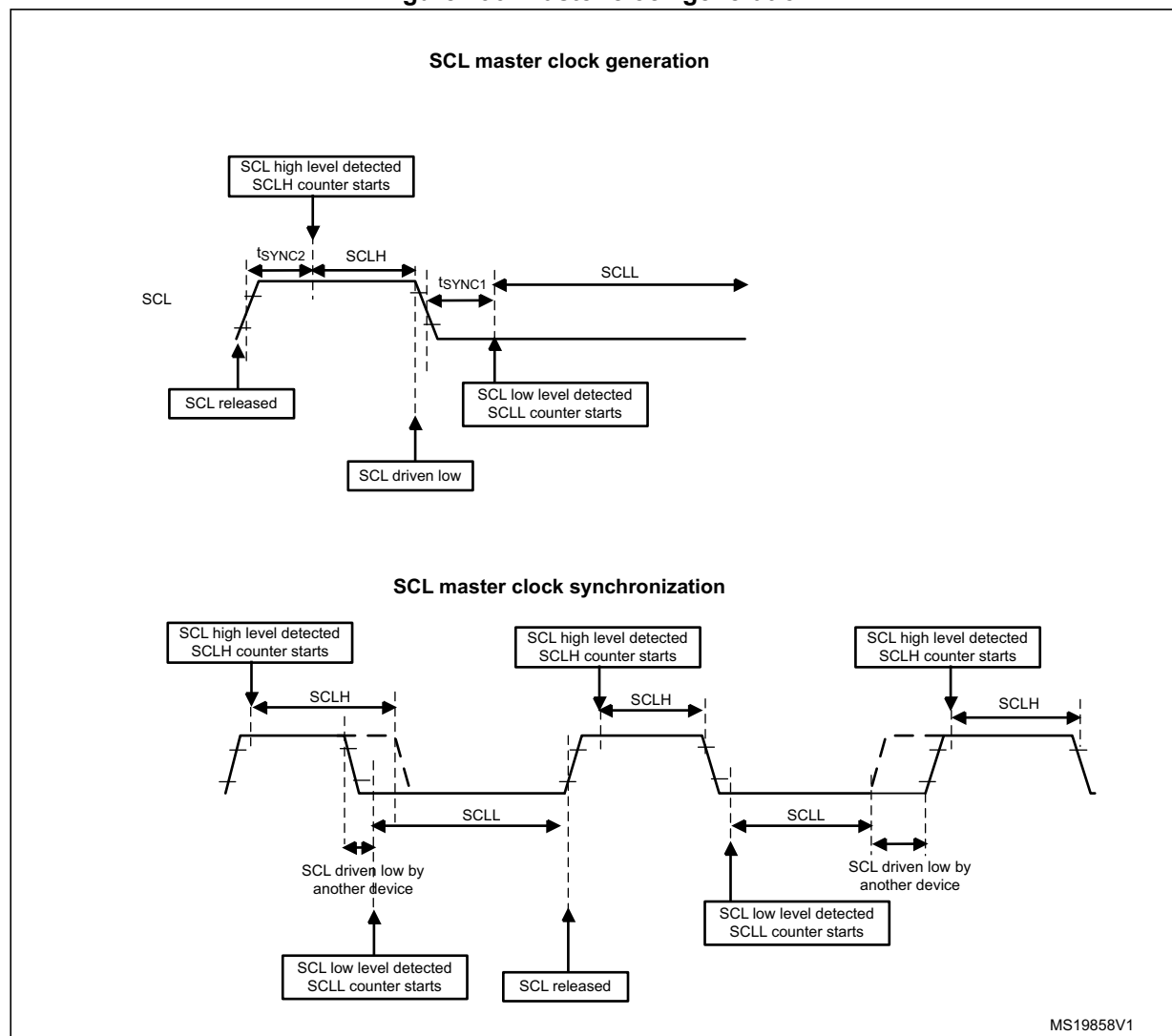
The duration of t_{SYNC1} depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: $\text{DNF} \times t_{\text{I2CCLK}}$
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

The duration of t_{SYNC2} depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: $\text{DNF} \times t_{\text{I2CCLK}}$
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

Figure 296. Master clock generation



Caution: In order to be I²C or SMBus compliant, the master clock must respect the timings given below:

Table 140. I²C-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_{SCL}	SCL clock frequency		100		400		1000		100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6		0.26	-	4.0	-	μs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6		0.26	-	4.7	-	μs
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6		0.26	-	4.0	-	μs

Table 140. I²C-SMBUS specification clock timings (continued)

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBUS		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3		0.5	-	4.7	-	μ s
t_{LOW}	Low period of the SCL clock	4.7	-	1.3		0.5	-	4.7	-	μ s
t_{HIGH}	Period of the SCL clock	4.0	-	0.6		0.26	-	4.0	50	μ s
t_r	Rise time of both SDA and SCL signals	-	1000	-	300		120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300		120	-	300	ns

Note: $SCLL$ is also used to generate the t_{BUF} and $t_{SU:STA}$ timings.

$SCLH$ is also used to generate the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to [Section 28.4.9: I2C_TIMINGR register configuration examples](#) for examples of I2C_TIMINGR settings vs. I2CCLK frequency.

Master communication initialization (address phase)

In order to initiate the communication, you must program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configured to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

You must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

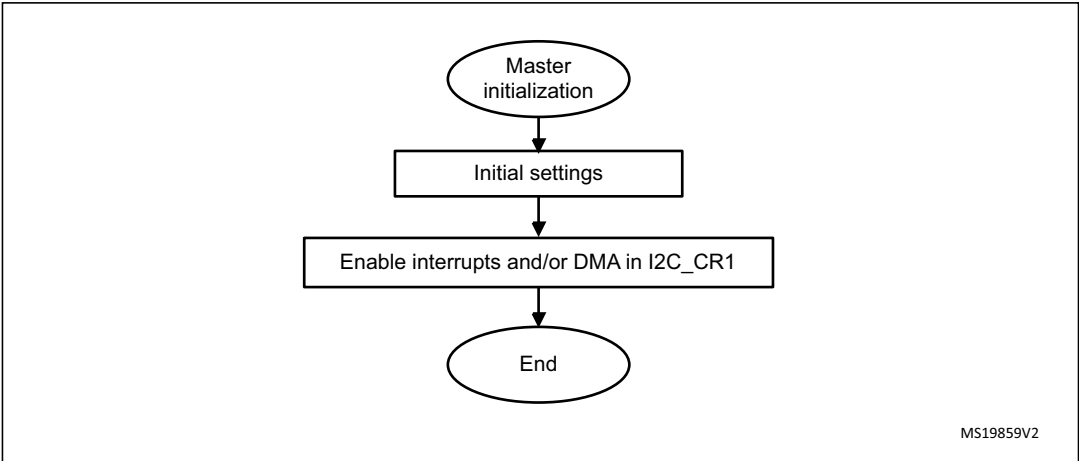
Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of t_{BUF} .

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

Note: The START bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs. If the I2C is addressed as a slave (ADDR=1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared when the ADDRCONF bit is set.

Note: The same procedure is applied for a Repeated Start condition. In this case BUSY=1.

Figure 297. Master initialization flowchart

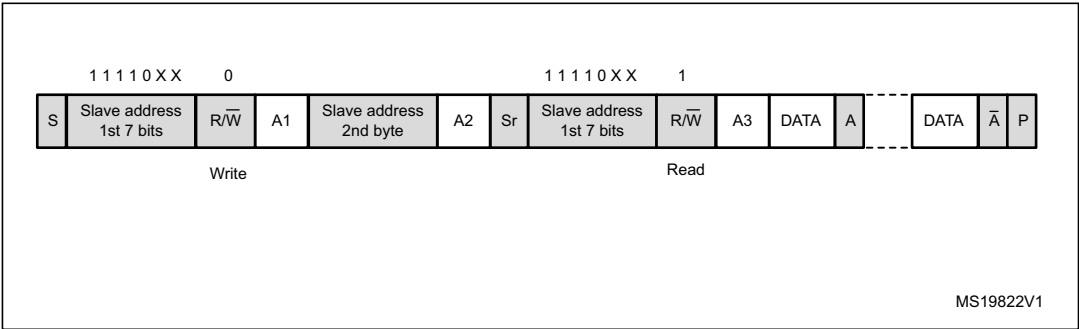


MS19859V2

Initialization of a master receiver addressing a 10-bit address slave

- If the slave address is in 10-bit format, you can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set: (Re)Start + Slave address 10-bit header Write + Slave address 2nd byte + REStart + Slave address 10-bit header Read

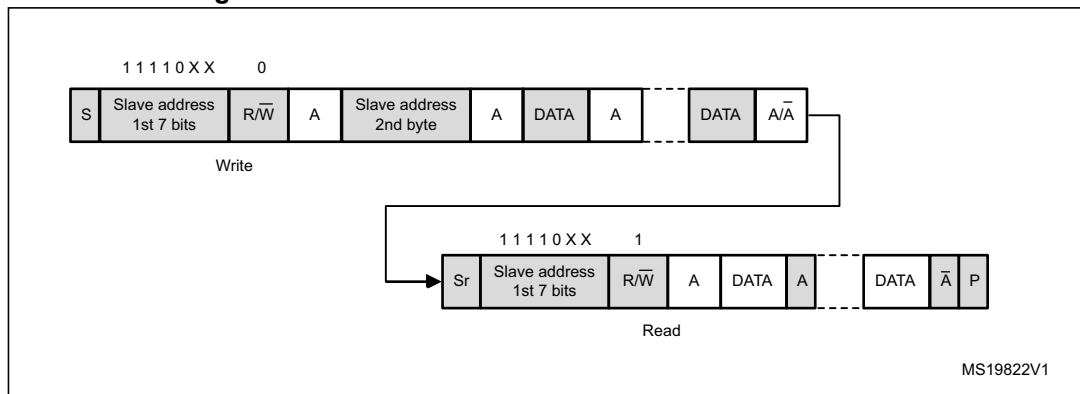
Figure 298. 10-bit address read access with HEAD10R=0



MS19822V1

- If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated start is set with the 10 bit slave address configured with HEAD10R=1. In this case the master sends this sequence: ReStart + Slave address 10-bit header Read

Figure 299. 10-bit address read access with HEAD10R=1



Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
 - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
 - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:

A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.

A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.

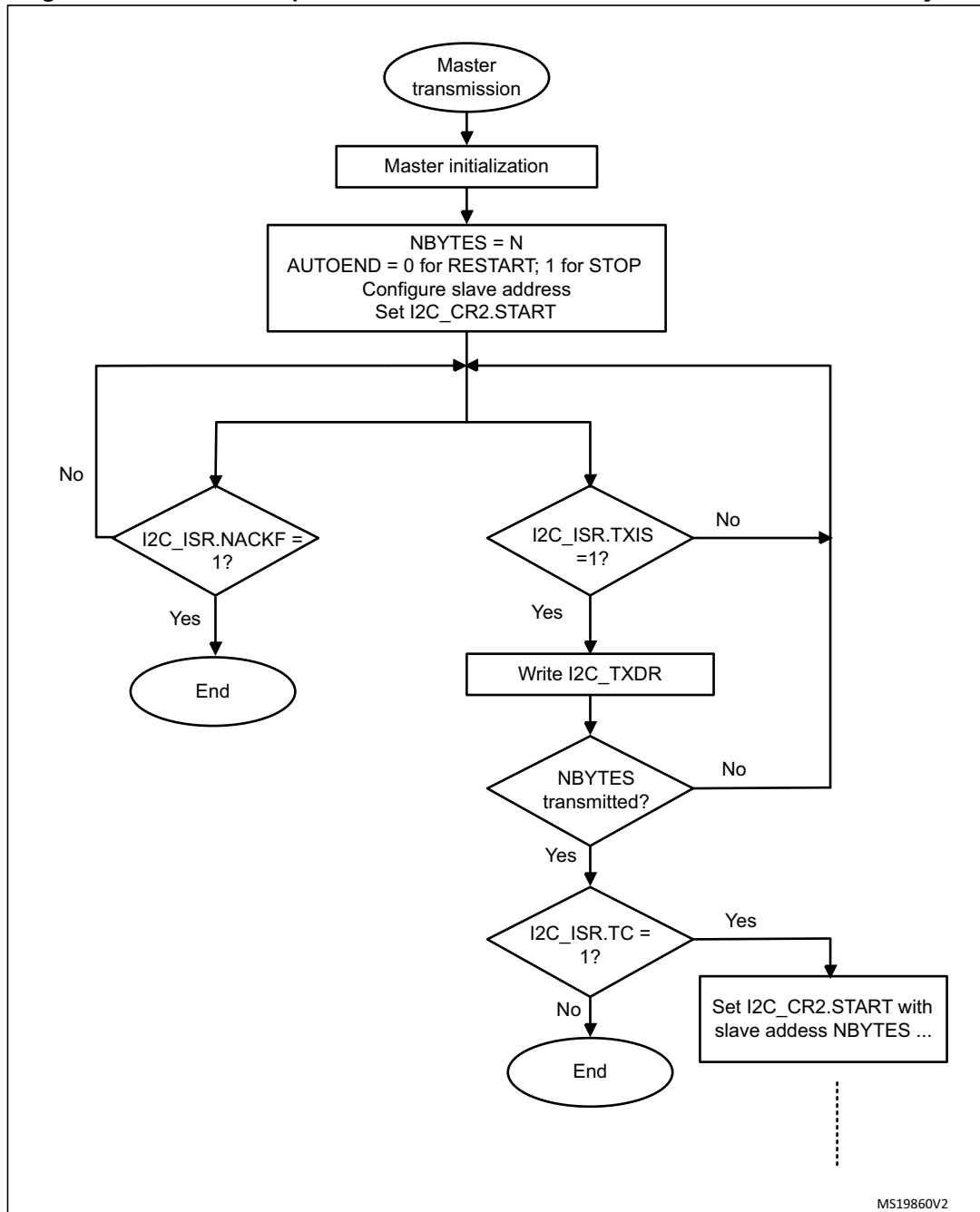
Figure 300. Transfer sequence flowchart for I2C master transmitter for $N \leq 255$ bytes

Figure 301. Transfer sequence flowchart for I2C master transmitter for N>255 bytes

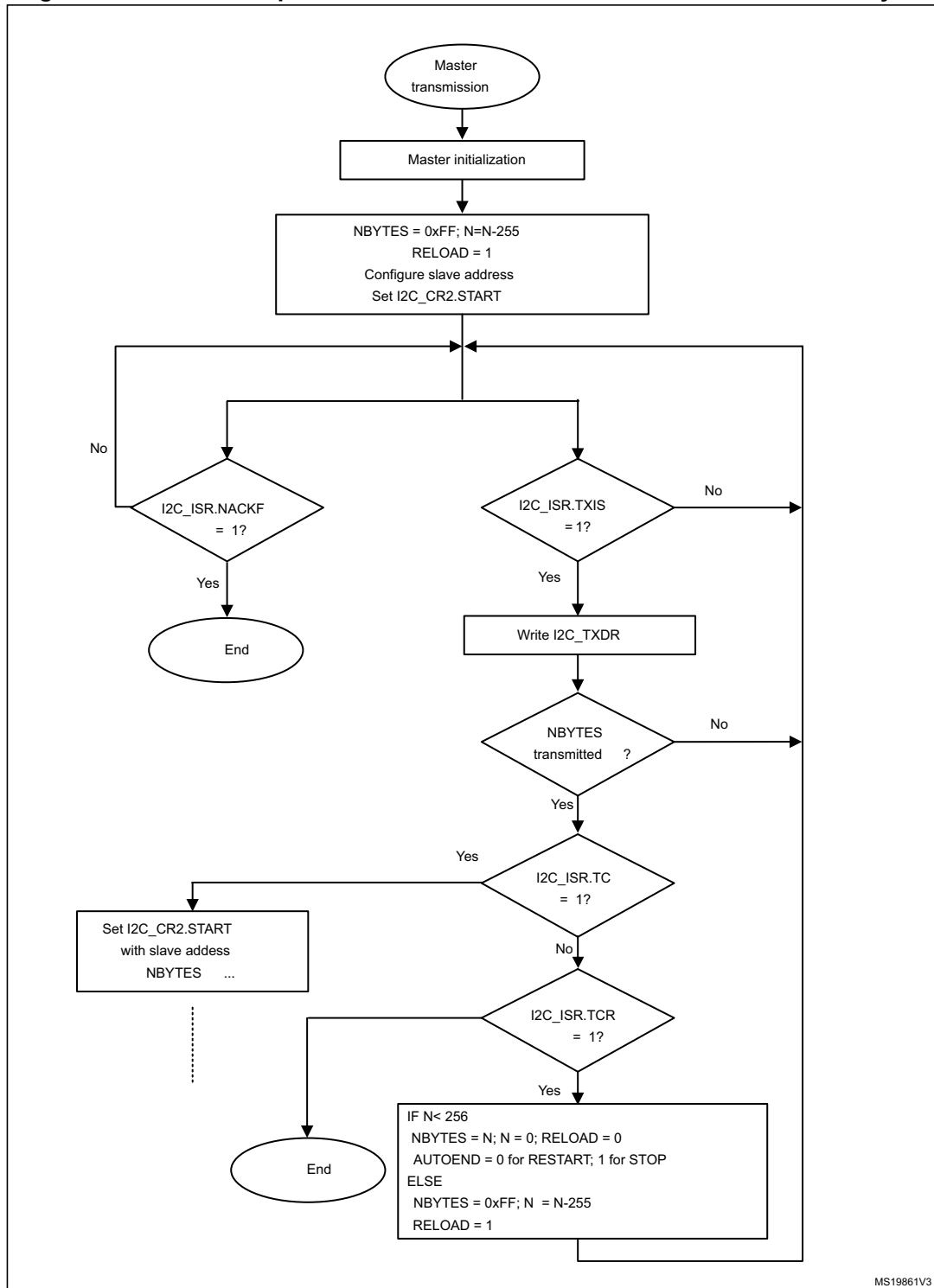
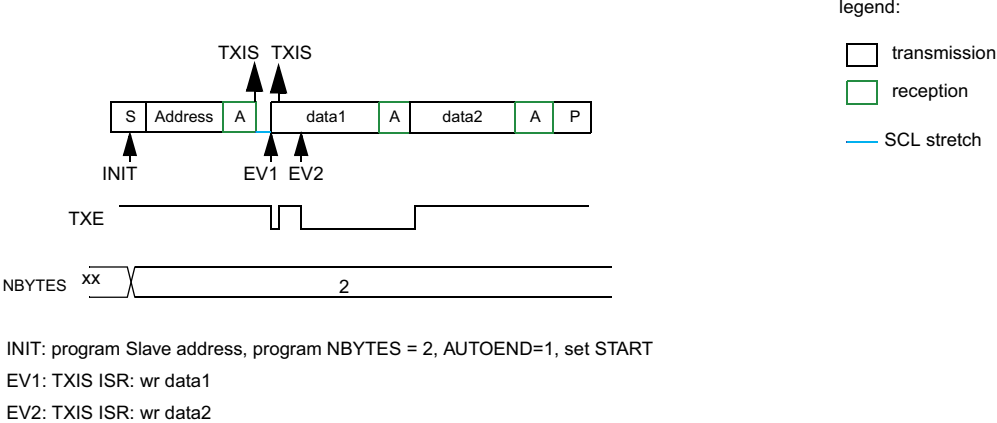
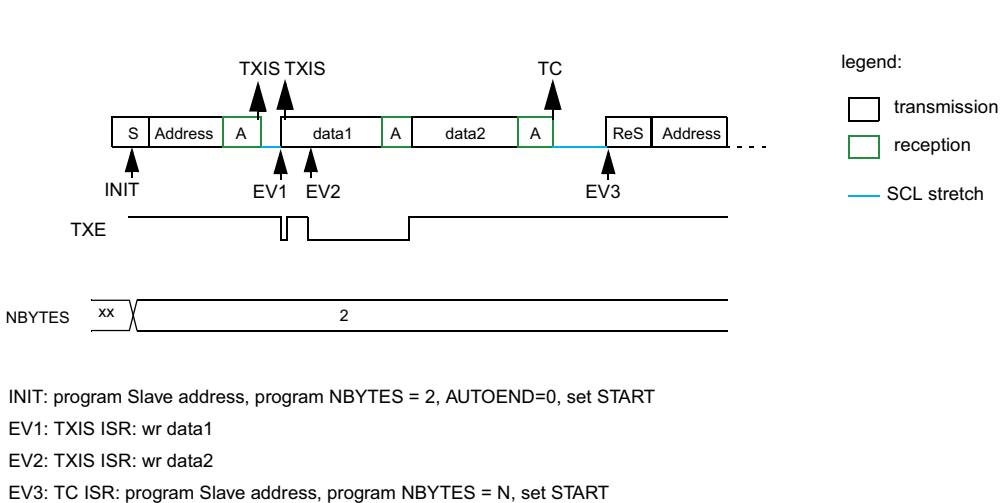


Figure 302. Transfer bus diagrams for I2C master transmitter

Example I2C master transmitter 2 bytes, automatic end mode (STOP)



Example I2C master transmitter 2 bytes, software end mode (RESTART)



MS19862V1

Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
 - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
 - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.

A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

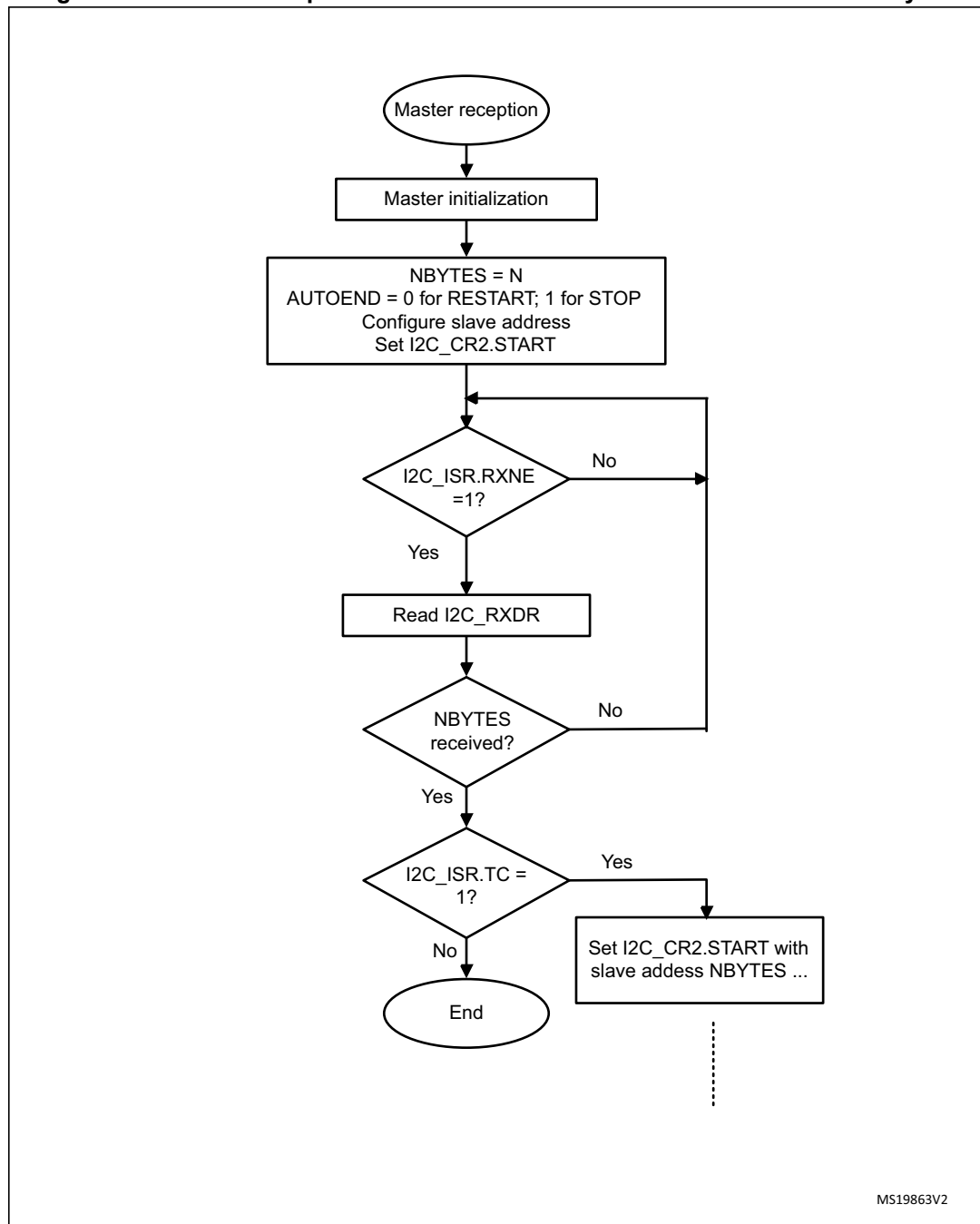
Figure 303. Transfer sequence flowchart for I2C master receiver for $N \leq 255$ bytes

Figure 304. Transfer sequence flowchart for I2C master receiver for N > 255 bytes

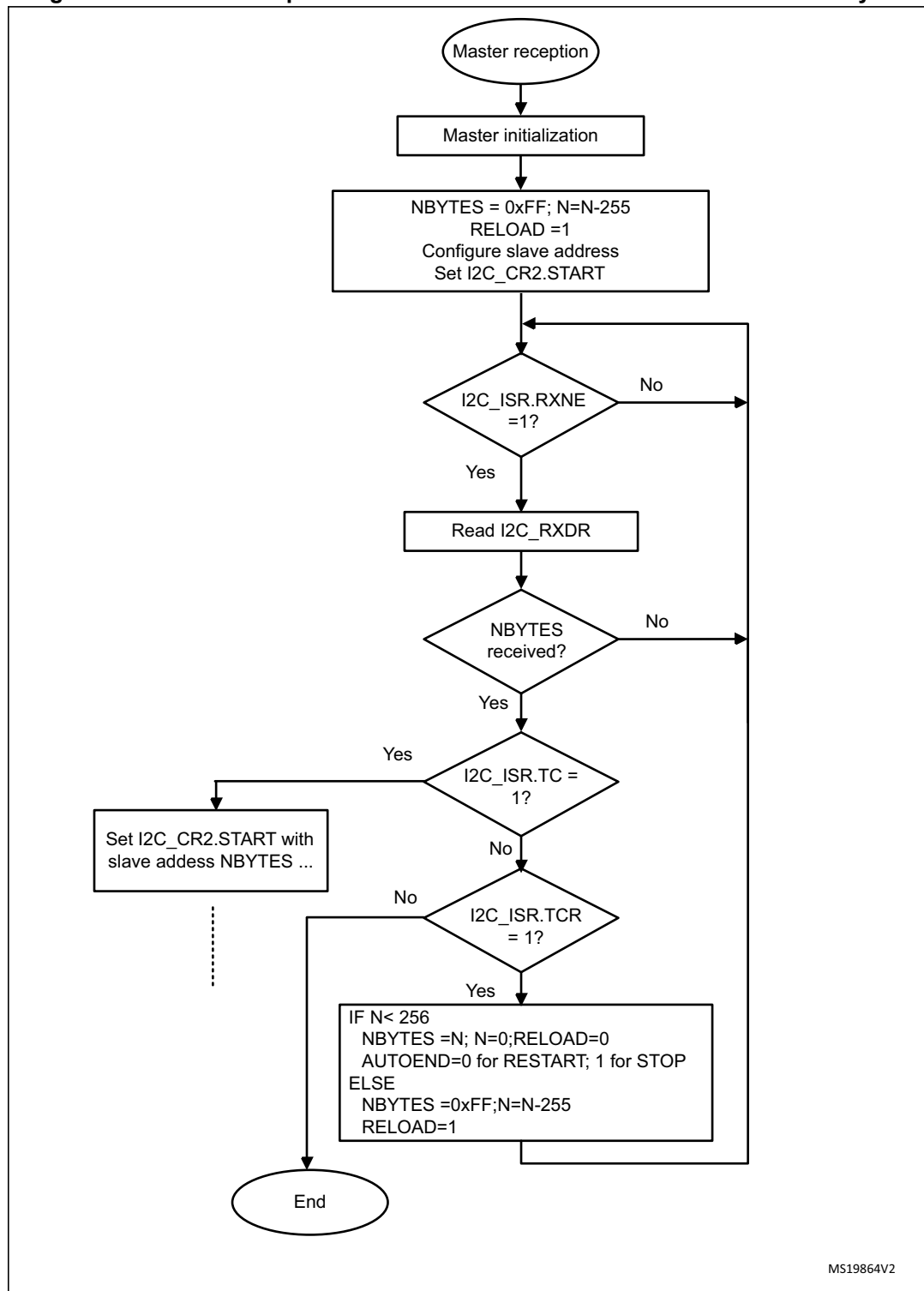
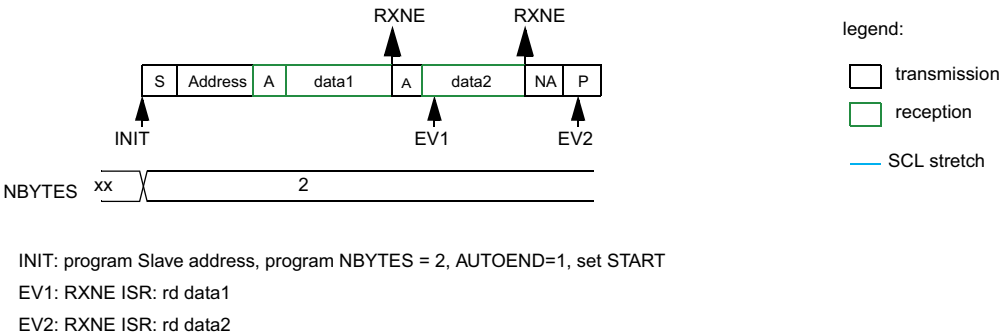
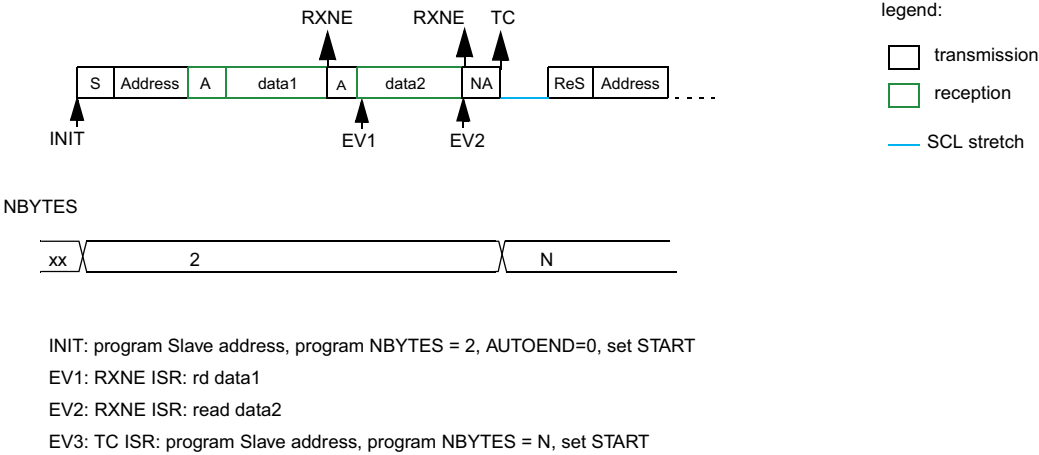


Figure 305. Transfer bus diagrams for I2C master receiver

Example I2C master receiver 2 bytes, automatic end mode (STOP)



Example I2C master receiver 2 bytes, software end mode (RESTART)



MS19865V1

28.4.9 I2C_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I²C specification. In order to get more accurate configuration values, please refer to application note AN4235 *I²C timing configuration tool* and the associated software STSW-STM32126.

Table 141. Examples of timings settings for $f_{I2CCLK} = 8\text{ MHz}$

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC	1	1	0	0
SCLL	0xC7	0x13	0x9	0x6
t_{SCLL}	200x250 ns = 50 μ s	20x250 ns = 5.0 μ s	10x125 ns = 1250 ns	7x125 ns = 875 ns
SCLH	0xC3	0xF	0x3	0x3
t_{SCLH}	196x250 ns = 49 μ s	16x250 ns = 4.0 μ s	4x125ns = 500ns	4x125 ns = 500 ns
$t_{SCL}^{(1)}$	~100 μ s ⁽²⁾	~10 μ s ⁽²⁾	~2500 ns ⁽³⁾	~2000 ns ⁽⁴⁾
SDADEL	0x2	0x2	0x1	0x0
t_{SDADEL}	2x250 ns = 500 ns	2x250 ns = 500 ns	1x125 ns = 125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5x250 ns = 1250 ns	5x250 ns = 1250 ns	4x125 ns = 500 ns	2x125 ns = 250 ns

1. SCL period t_{SCL} is greater than $t_{SCLL} + t_{SCLH}$ due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 500\text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 1000\text{ ns}$
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 500\text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 750\text{ ns}$
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 500\text{ ns}$. Example with $t_{SYNC1} + t_{SYNC2} = 655\text{ ns}$

Table 142. Examples of timings settings for $f_{I2CCLK} = 16\text{ MHz}$

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 μ s	20 x 250 ns = 5.0 μ s	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 μ s	16 x 250 ns = 4.0 μ s	4 x 125ns = 500 ns	3 x 62.5 ns = 187.5 ns
$t_{SCL}^{(1)}$	~100 μ s ⁽²⁾	~10 μ s ⁽²⁾	~2500 ns ⁽³⁾	~1000 ns ⁽⁴⁾
SDADEL	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. SCL period t_{SCL} is greater than $t_{SCLL} + t_{SCLH}$ due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 250$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 1000$ ns
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 250$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 750$ ns
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 250$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 500$ ns

Table 143. Examples of timings settings for $f_{I2CCLK} = 48$ MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	0xB	0xB	5	5
SCLL	0xC7	0x13	0x9	0x3
t_{SCLL}	200×250 ns = 50 μ s	20×250 ns = 5.0 μ s	10×125 ns = 1250 ns	4×125 ns = 500 ns
SCLH	0xC3	0xF	0x3	0x1
t_{SCLH}	196×250 ns = 49 μ s	16×250 ns = 4.0 μ s	4×125 ns = 500 ns	2×125 ns = 250 ns
$t_{SCL}^{(1)}$	$\sim 100 \mu$ s ⁽²⁾	$\sim 10 \mu$ s ⁽²⁾	~ 2500 ns ⁽³⁾	~ 875 ns ⁽⁴⁾
SDADEL	0x2	0x2	0x3	0x0
t_{SDADEL}	2×250 ns = 500 ns	2×250 ns = 500 ns	3×125 ns = 375 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5×250 ns = 1250 ns	5×250 ns = 1250 ns	4×125 ns = 500 ns	2×125 ns = 250 ns

1. The SCL period t_{SCL} is greater than $t_{SCLL} + t_{SCLH}$ due to the SCL internal detection delay. Values provided for t_{SCL} are only examples.
2. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 83.3$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 1000$ ns
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 83.3$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 750$ ns
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is $4 \times t_{I2CCLK} = 83.3$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 250$ ns

28.4.10 SMBus specific features

This section is relevant only when SMBus feature is supported. Please refer to [Section 28.3: 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 I²C principles of operation. SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBUS specification rev 2.0 (<http://smbus.org>).

The System Management Bus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.

SMBUS is based on I²C specification rev 2.1.

Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software.

For more details of these protocols, refer to SMBus specification version 2.0 (<http://smbus.org>).

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands should be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus Address Resolution Protocol, refer to SMBus specification version 2.0 (<http://smbus.org>).

Received Command and Data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to [Slave Byte Control Mode on page 780](#) section for more details.

Host Notify protocol

This peripheral supports the Host Notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host will acknowledge the SMBus Host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and

simultaneously accesses all SMBALERT# devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled SMBALERT# low will acknowledge the Alert Response Address.

When configured as a slave device (SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0.

Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x_8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows to send a Not Acknowledge automatically when the received byte does not match with the hardware calculated PEC.

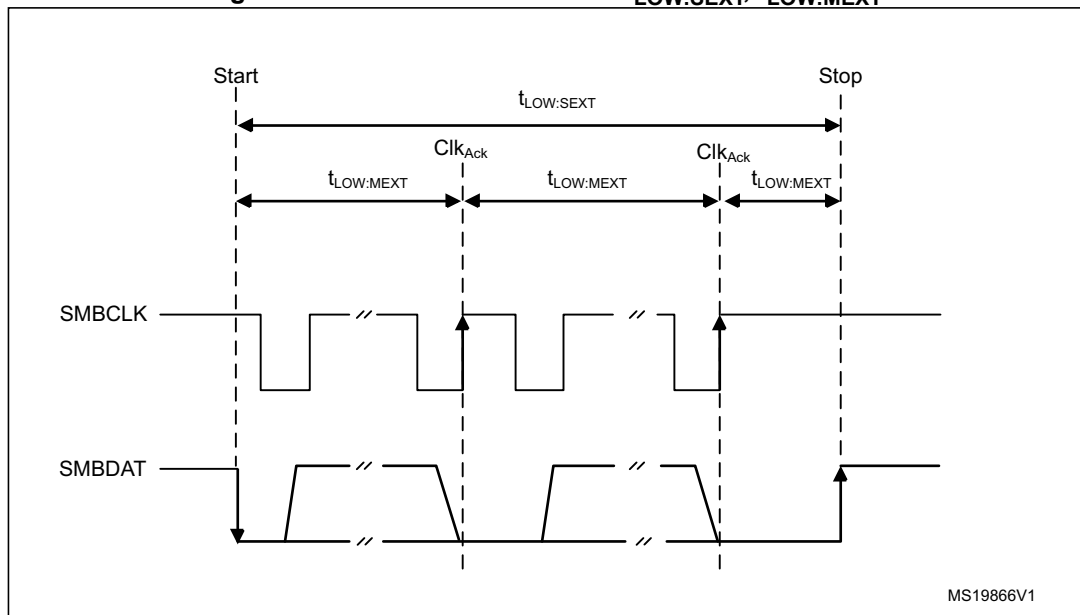
Timeouts

This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification version 2.0.

Table 144. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
t_{TIMEOUT}	Detect clock low timeout	25	35	ms
$t_{\text{LOW:SEXT}}^{(1)}$	Cumulative clock low extend time (slave device)	-	25	ms
$t_{\text{LOW:MEXT}}^{(2)}$	Cumulative clock low extend time (master device)	-	10	ms

1. $t_{\text{LOW:SEXT}}$ is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another slave device or the master will also extend the clock causing the combined clock low extend time to be greater than $t_{\text{LOW:SEXT}}$. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.
2. $t_{\text{LOW:MEXT}}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master will also extend the clock causing the combined clock low time to be greater than $t_{\text{LOW:MEXT}}$ on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

Figure 306. Timeout intervals for $t_{\text{LOW:SEXT}}$, $t_{\text{LOW:MEXT}}$ 

Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for t_{IDLE} greater than $t_{\text{HIGH,MAX}}$. (refer to [Table 140: I2C-SMBUS specification clock timings](#))

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

28.4.11 SMBus initialization

This section is relevant only when SMBus feature is supported. Please refer to [Section 28.3: I2C implementation](#).

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

Received Command and Data Acknowledge control (Slave mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to [Slave Byte Control Mode on page 780](#) for more details.

Specific address (Slave mode)

The specific SMBus addresses should be enabled if needed. Refer to [Bus idle detection on page 804](#) for more details.

- The SMBus Device Default address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C_CR1 register.
- The SMBus Host address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C_CR1 register.
- The Alert Response Address (0b0001100) is enabled by setting the ALERTEN bit in the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTES-1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C is enabled.

Table 145. SMBUS with PEC configuration table

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Master Tx/Rx NBYTES + PEC+ STOP	x	0	1	1
Master Tx/Rx NBYTES + PEC + ReSTART	x	0	0	1
Slave Tx/Rx with PEC	1	0	x	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the 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 version 2.0.

- t_{TIMEOUT} check
In order to enable the t_{TIMEOUT} check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the t_{TIMEOUT} parameter. The TIDLE bit must be configured to '0' in order to detect the SCL low level timeout.
Then the timer is enabled by setting the TIMOUTEN in the I2C_TIMEOUTR register.
If SCL is tied low for a time greater than $(\text{TIMEOUTA}+1) \times 2048 \times t_{\text{I2CCLK}}$, the TIMEOUT flag is set in the I2C_ISR register.
Refer to [Table 146: Examples of TIMEOUTA settings for various I2CCLK frequencies \(max \$t_{\text{TIMEOUT}} = 25 \text{ ms}\$ \)](#).

Caution: Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMEOUTEN bit is set.

- $t_{\text{LOW:SEXT}}$ and $t_{\text{LOW:MEXT}}$ check

Depending on if the peripheral is configured as a master or as a slave, The 12-bit TIMEOUTB timer must be configured in order to check $t_{\text{LOW:SEXT}}$ for a slave and $t_{\text{LOW:MEXT}}$ for a master. As the standard specifies only a maximum, you can choose the same value for the both.

Then the timer is enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for a time greater than $(\text{TIMEOUTB}+1) \times 2048 \times t_{\text{I2CCLK}}$, and in the timeout interval described in [Bus idle detection on page 804](#) section, the TIMEOUT flag is set in the I2C_ISR register.

Refer to [Table 147: Examples of TIMEOUTB settings for various I2CCLK frequencies](#)

Caution: Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

Bus Idle detection

In order to enable the t_{IDLE} check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the t_{IDLE} parameter. The TIDLE bit must be configured to '1' in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMOUTEN bit in the I2C_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than $(\text{TIMEOUTA}+1) \times 4 \times t_{\text{I2CCLK}}$, the TIMEOUT flag is set in the I2C_ISR register.

Refer to [Table 148: Examples of TIMEOUTA settings for various I2CCLK frequencies \(max tIDLE = 50 \$\mu\$ s\)](#)

Caution: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

28.4.12 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Please refer to [Section 28.3: I2C implementation](#).

- Configuring the maximum duration of t_{TIMEOUT} to 25 ms:

Table 146. Examples of TIMEOUTA settings for various I2CCLK frequencies (max t_{TIMEOUT} = 25 ms)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	t_{TIMEOUT}
8 MHz	0x61	0	1	$98 \times 2048 \times 125 \text{ ns} = 25 \text{ ms}$
16 MHz	0xC3	0	1	$196 \times 2048 \times 62.5 \text{ ns} = 25 \text{ ms}$
48 MHz	0x249	0	1	$586 \times 2048 \times 20.08 \text{ ns} = 25 \text{ ms}$

- Configuring the maximum duration of $t_{\text{LOW:SEXT}}$ and $t_{\text{LOW:MEXT}}$ to 8 ms:

Table 147. Examples of TIMEOUTB settings for various I2CCLK frequencies

f_{I2CCLK}	TIMEOUTB[11:0] bits	TEXTEN bit	$t_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

- Configuring the maximum duration of t_{IDLE} to 50 μ s

Table 148. Examples of TIMEOUTA settings for various I2CCLK frequencies
(max t_{IDLE} = 50 μ s)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	t_{TIDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 μ s
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 μ s
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 μ s

28.4.13 SMBus slave mode

This section is relevant only when SMBus feature is supported. Please refer to [Section 28.3: I2C implementation](#).

In addition to 2C slave transfer management (refer to [Section 28.4.7: I2C slave mode](#)) some additional software flowcharts are provided to support SMBus.

SMBus Slave transmitter

When the IP is used in SMBus, SBC must be programmed to '1' in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts will be NBYTES-1 and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES-1 data transfer.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 307. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC

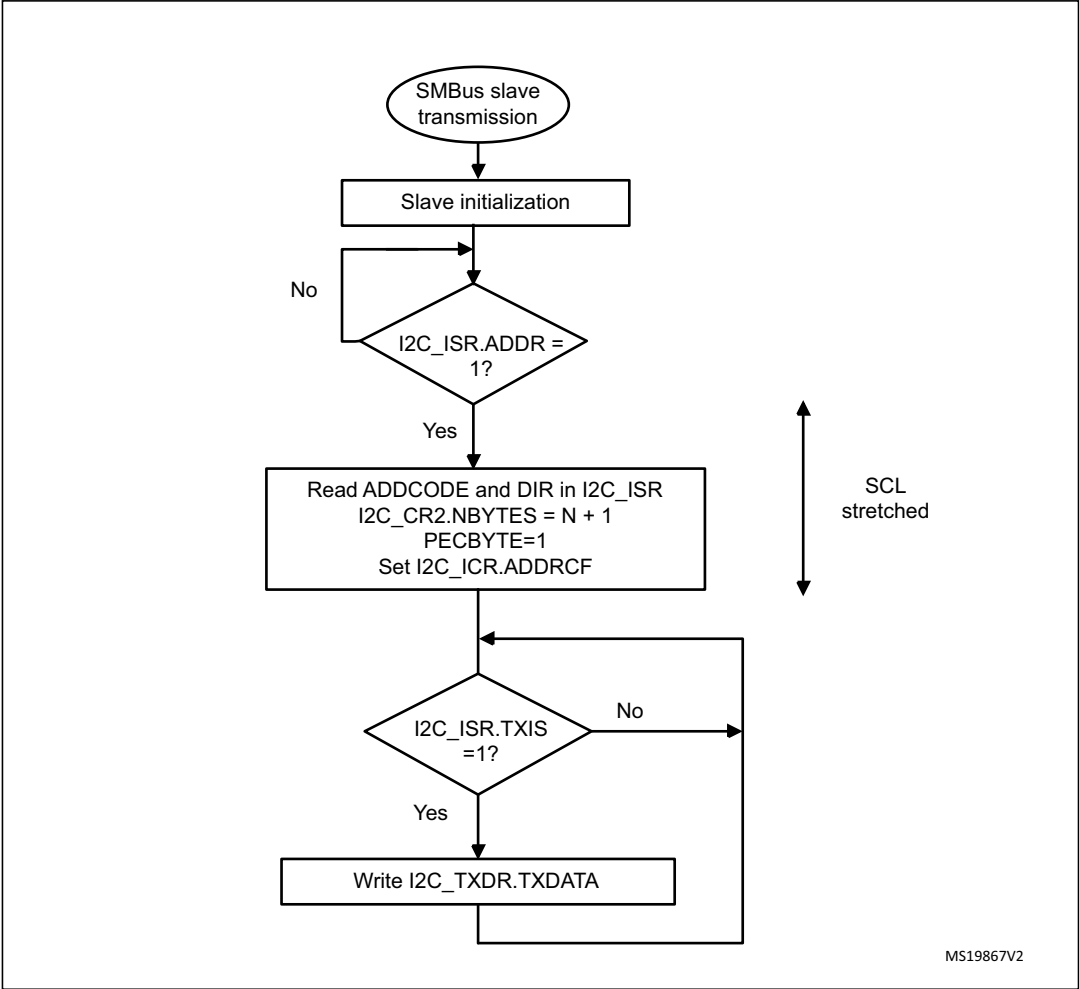
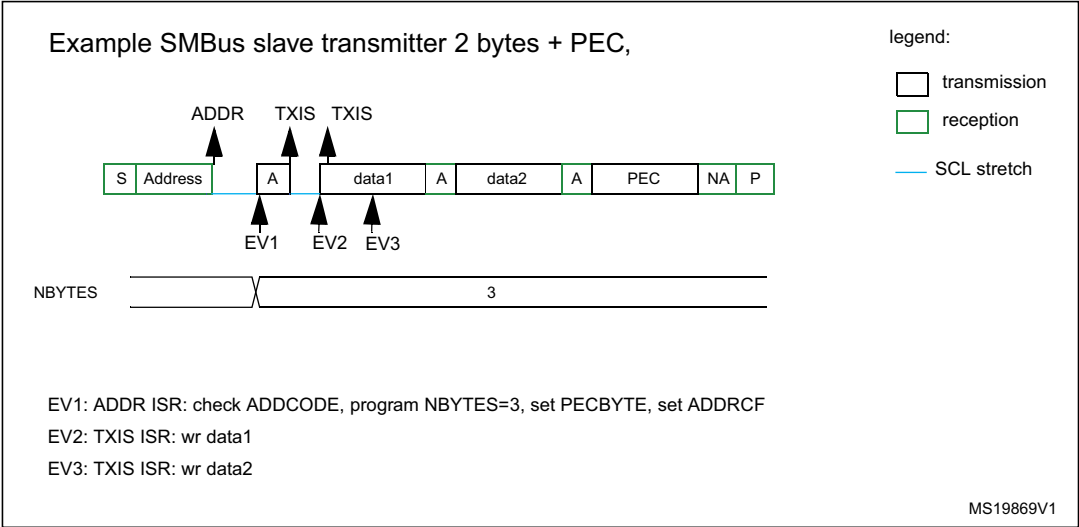


Figure 308. Transfer bus diagrams for SMBus slave transmitter (SBC=1)



SMBus Slave receiver

When the I2C is used in SMBus mode, SBC must be programmed to '1' in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to [Slave Byte Control Mode on page 780](#) for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES-1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, you can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES-1 are received, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 309. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC

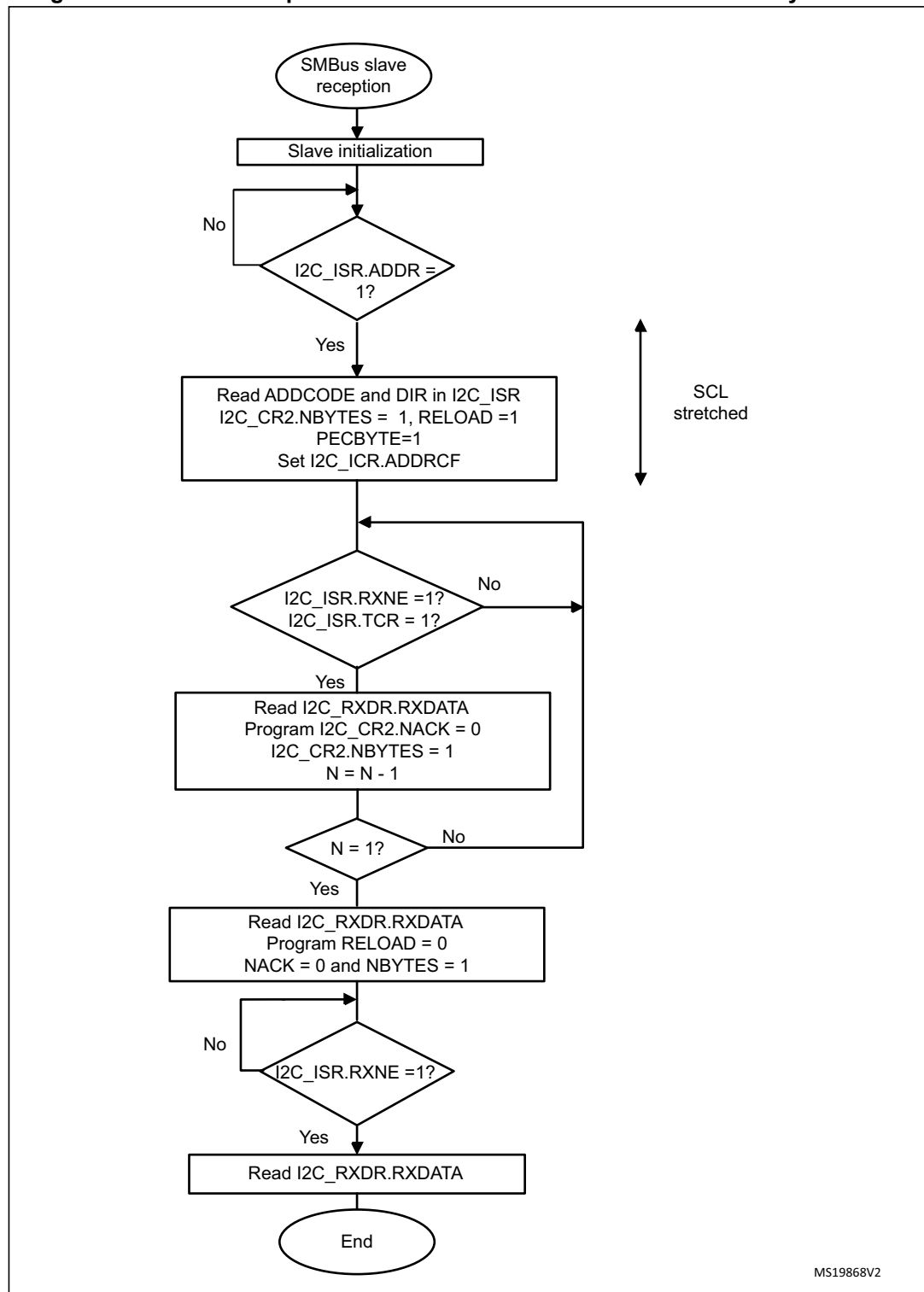
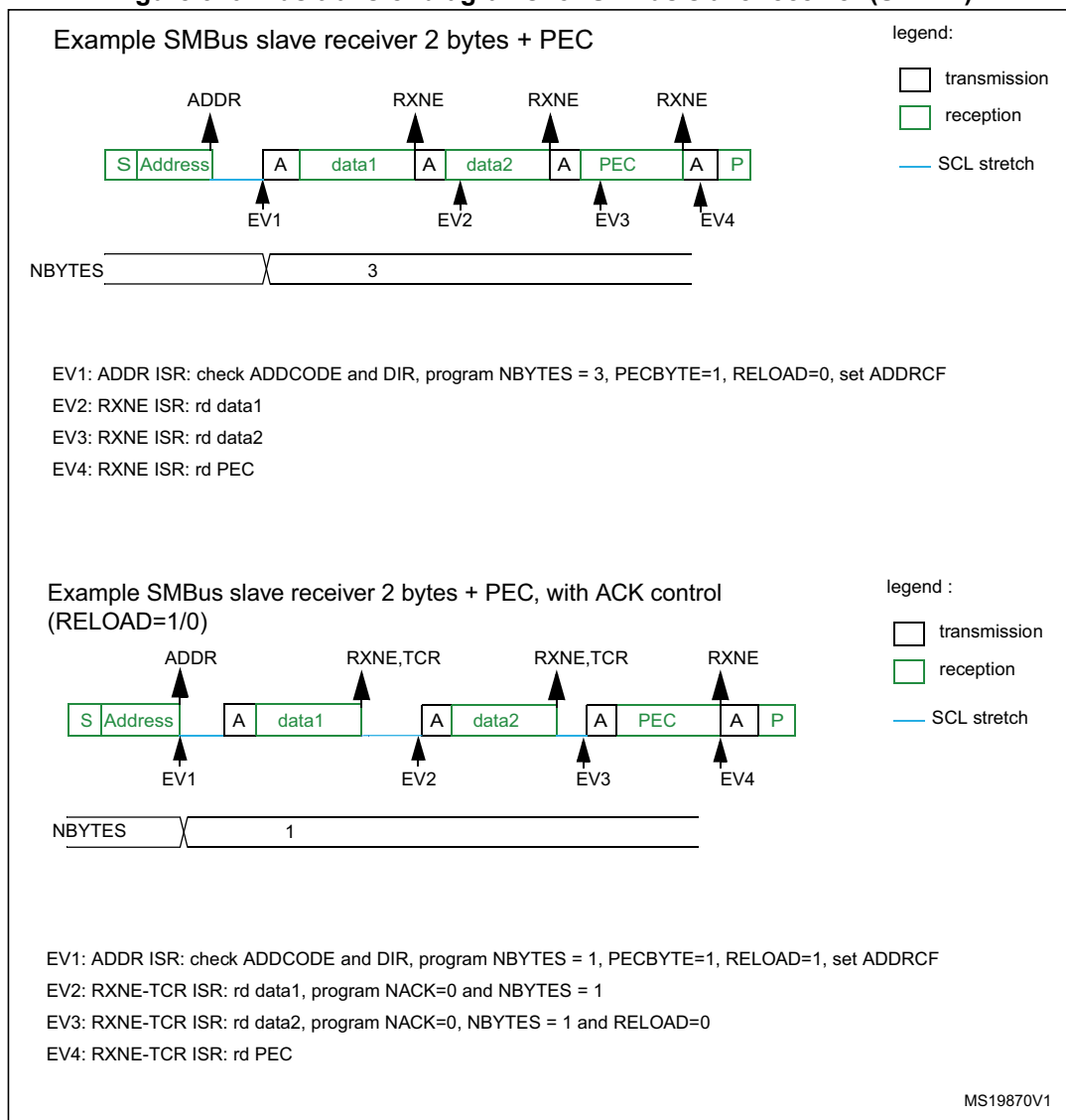


Figure 310. Bus transfer diagrams for SMBus slave receiver (SBC=1)

This section is relevant only when SMBus feature is supported. Please refer to [Section 28.3: I2C implementation](#).

In addition to I2C master transfer management (refer to [Section 28.4.8: I2C master mode](#)) some additional software flowcharts are provided to support SMBus.

SMBus Master transmitter

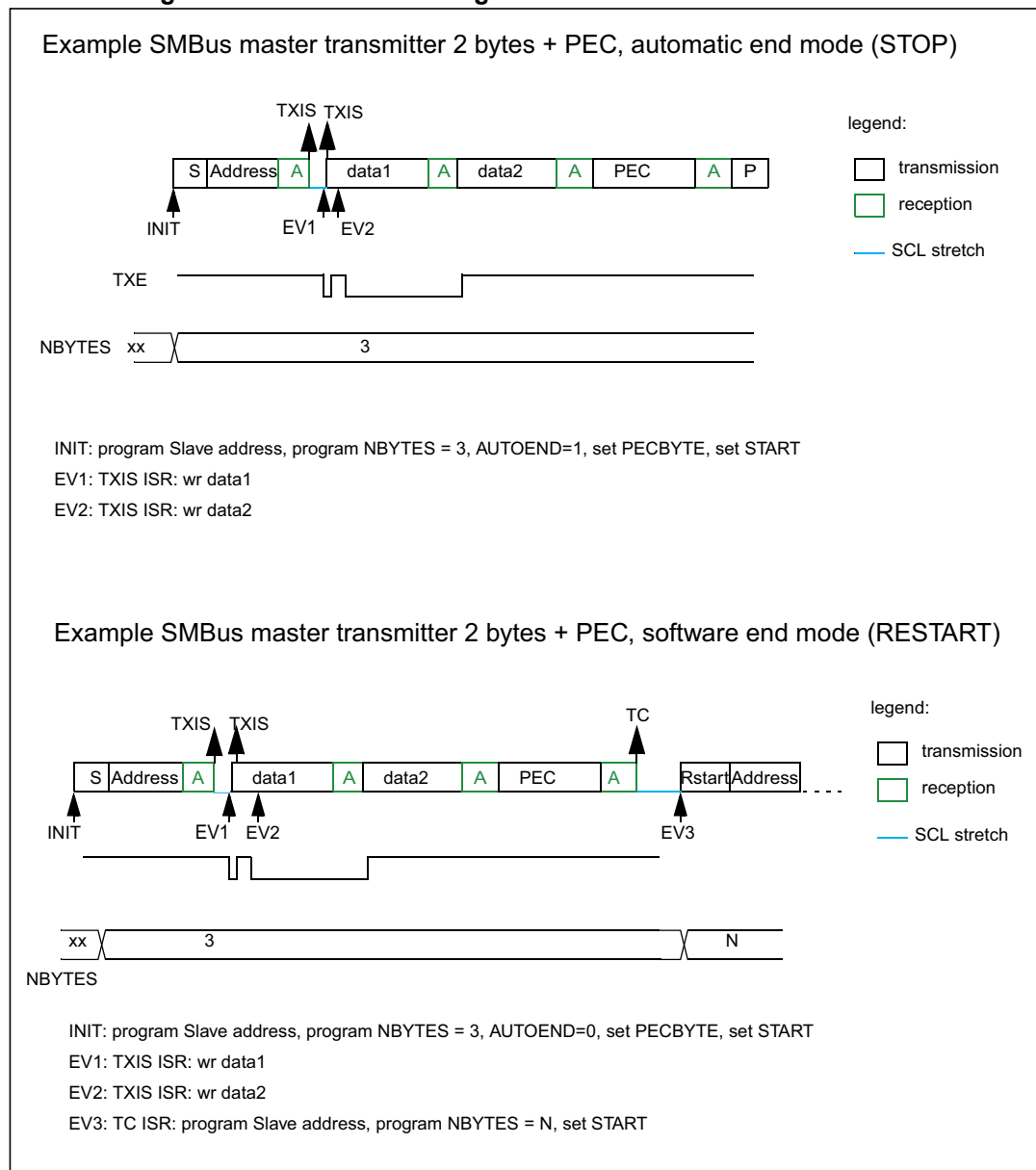
When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts will be NBYTES-1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode should be selected (AUTOEND=1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES-1 have been transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 311. Bus transfer diagrams for SMBus master transmitter



SMBus Master receiver

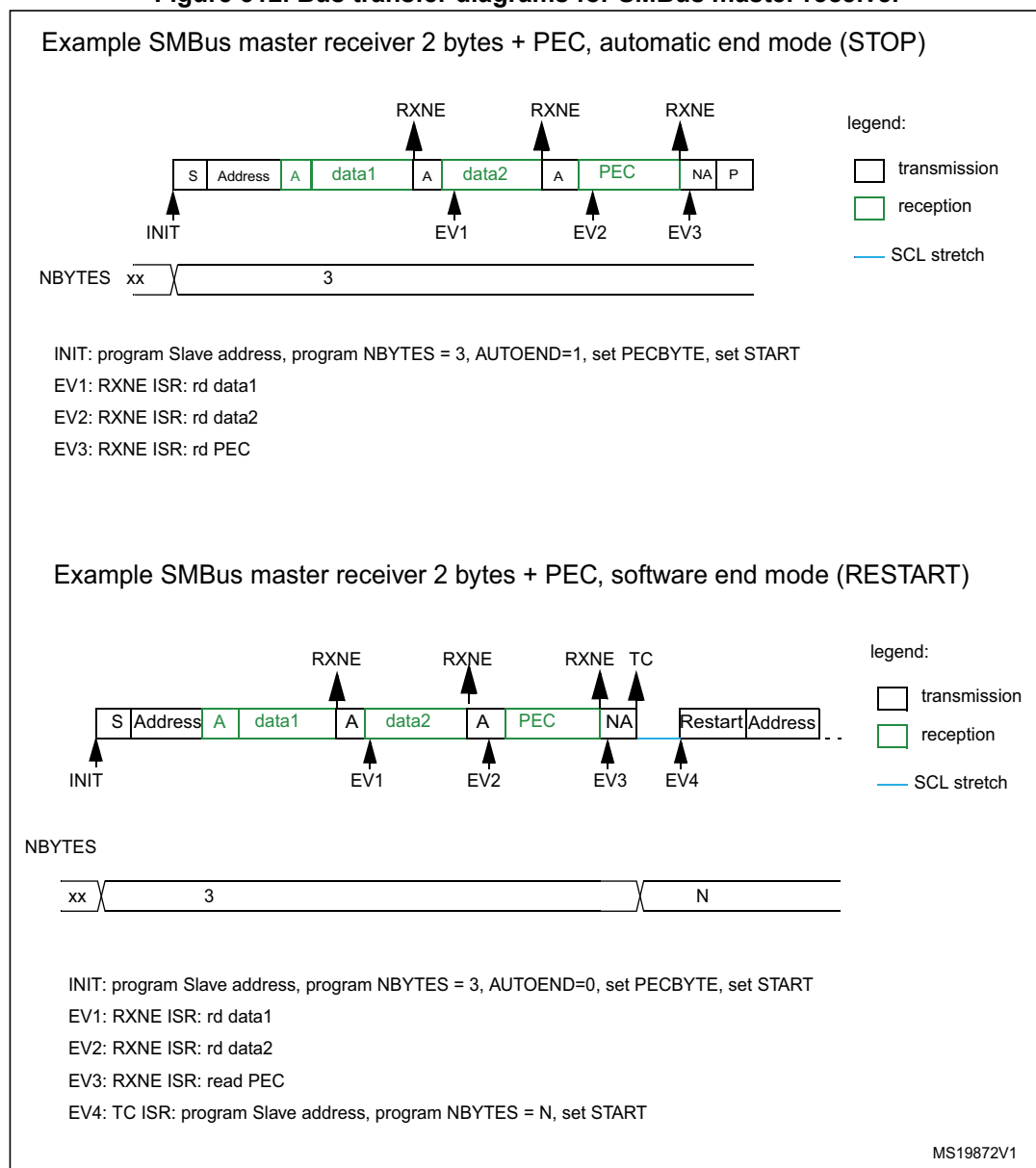
When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND=1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case,

after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 312. Bus transfer diagrams for SMBus master receiver



28.4.14 Wakeup from Stop mode on address match

This section is relevant only when Wakeup from Stop mode feature is supported. Please refer to [Section 28.3: I2C implementation](#).

The I2C is able to wakeup the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wakeup from Stop mode is enabled by setting the WUPEN bit in the I2C_CR1 register. The HSI oscillator must be selected as the clock source for I2CCLK in order to allow wakeup from Stop mode.

During Stop mode, the HSI is switched off. When a START is detected, the I2C interface switches the HSI on, and stretches SCL low until HSI is woken up.

HSI is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wakeup time. The stretch is released when ADDR flag is cleared by software, and the transfer goes on normally.

If the address does not match, the HSI is switched off again and the MCU is not woken up.

Note: *If the I2C clock is the system clock, or if WUPEN = 0, the HSI oscillator is not switched on after a START is received.*

Only an ADDR interrupt can wakeup the MCU. Therefore do not enter Stop mode when the I2C is performing a transfer as a master, or as an addressed slave after the ADDR flag is set. This can be managed by clearing SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wakeup from Stop mode feature. If the DNF bit is not equal to 0, setting the WUPEN bit has no effect.

Caution: This feature is available only when the I2C clock source is the HSI oscillator.

Caution: Clock stretching must be enabled (NOSTRETCH=0) to ensure proper operation of the wakeup from Stop mode feature.

Caution: If wakeup from Stop mode is disabled (WUPEN=0), the I2C peripheral must be disabled before entering Stop mode (PE=0).

28.4.15 Error conditions

The following are the error conditions which may cause communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in slave mode when NOSTRETCH=1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
 - When STOPF=1 and the first data byte should be sent. The content of the I2C_TXDR register is sent if TXE=0, 0xFF if not.
 - When a new byte should be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Packet Error Checking Error (PECERR)

This section is relevant only when the SMBus feature is supported. Please refer to [Section 28.3: I2C implementation](#).

A PEC error is detected when the received PEC byte does not match with the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Timeout Error (TIMEOUT)

This section is relevant only when the SMBus feature is supported. Please refer to [Section 28.3: I2C implementation](#).

A timeout error occurs for any of these conditions:

- TIDLE=0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE=1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus $t_{\text{LOW:MEXT}}$ parameter)
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus $t_{\text{LOW:SEXT}}$ parameter)

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Alert (ALERT)

This section is relevant only when the SMBus feature is supported. Please refer to [Section 28.3: I2C implementation](#).

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN=1), the alert pin detection is enabled (ALERTEN=1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

28.4.16 DMA requests

Transmission using DMA

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see [Section 13: Direct memory access controller \(DMA\) on page 244](#)) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to [Master transmitter on page 792](#).
- In slave mode:
 - With NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
 - With NOSTRETCH=1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to [SMBus Slave transmitter on page 807](#) and [SMBus Master transmitter on page 811](#).

Note: If DMA is used for transmission, the TXIE bit does not need to be enabled.

Reception using DMA

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to [Section 13: Direct memory access controller](#)

(DMA) on page 244) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.
- In slave mode with NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see [Section 28.3: I2C implementation](#)): the PEC transfer is managed with the NBYTES counter. Refer to [SMBus Slave receiver on page 809](#) and [SMBus Master receiver on page 812](#).

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

28.4.17 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_SMBUS_TIMEOUT configuration bits in the DBG module.

28.5 I2C low-power modes

Table 149. low-power modes

Mode	Description
Sleep	No effect I2C interrupts cause the device to exit the Sleep mode.
Stop	The I2C registers content is kept. If WUPEN=1: the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. If WUPEN=0: the I2C must be disabled before entering Stop mode.
Standby	The I2C peripheral is powered down and must be reinitialized after exiting Standby.

28.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

Table 150. I2C Interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Receive buffer not empty	RXNE	Read I2C_RXDR register	RXIE
Transmit buffer interrupt status	TXIS	Write I2C_TXDR register	TXIE
Stop detection interrupt flag	STOPF	Write STOPCF=1	STOPIE

Table 150. I2C Interrupt requests (continued)

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Transfer Complete Reload	TCR	Write I2C_CR2 with NBYTES[7:0] \neq 0	TCIE
Transfer complete	TC	Write START=1 or STOP=1	
Address matched	ADDR	Write ADDRCONF=1	ADDRIE
NACK reception	NACKF	Write NACKCONF=1	NACKIE
Bus error	BERR	Write BERRCONF=1	ERRIE
Arbitration loss	ARLO	Write ARLOCONF=1	
Overrun/Underrun	OVR	Write OVRCONF=1	
PEC error	PECERR	Write PECERRCONF=1	
Timeout/t _{LOW} error	TIMEOUT	Write TIMEOUTCONF=1	
SMBus Alert	ALERT	Write ALERTCONF=1	

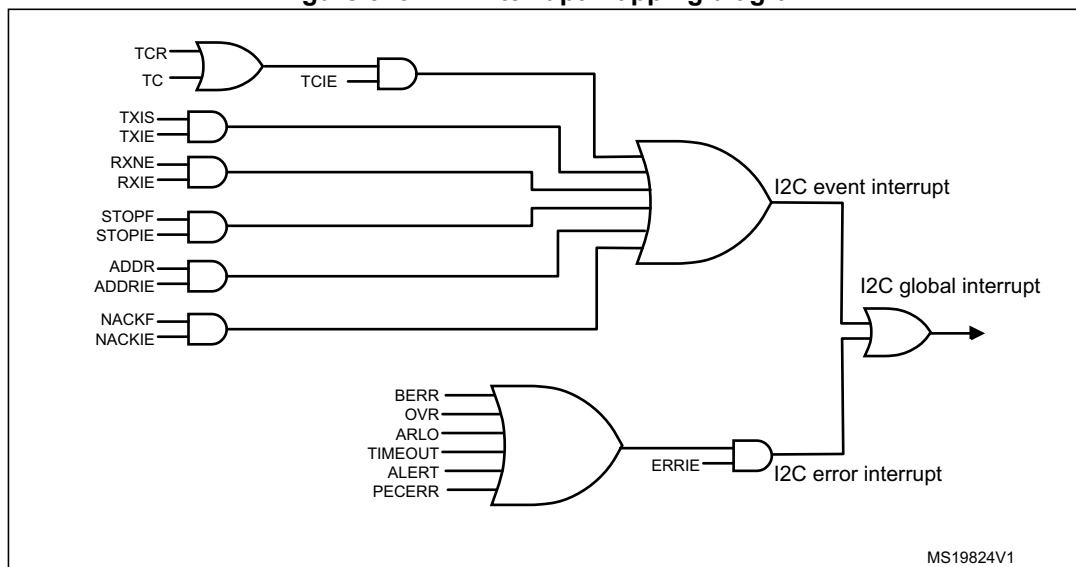
Depending on the product implementation, all these interrupts events can either share the same interrupt vector (I2C global interrupt), or be grouped into 2 interrupt vectors (I2C event interrupt and I2C error interrupt). Refer to [Table 77: STM32F302xB/C/D/E vector table](#) and [Table 78: STM32F302x6/8 vector table](#) for details.

In order to enable the I2C interrupts, the following sequence is required:

1. Configure and enable the I2C IRQ channel in the NVIC.
2. Configure the I2C to generate interrupts.

The I2C wakeup event is connected to the EXTI controller (refer to [Section 14.2: Extended interrupts and events controller \(EXTI\)](#)).

Figure 313. I2C interrupt mapping diagram



28.7 I2C registers

Refer to [Section 2.1 on page 42](#) for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

28.7.1 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 to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERT EN	SMBD EN	SMBH EN	GCEN	WUPE N	NOSTR ETCH	SBC
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMA EN	TXDMA EN	Res.	ANF OFF	DNF				ERRIE	TCIE	STOP IE	NACK IE	ADDR IE	RXIE	TXIE	PE
rw	rw		rw	rw				rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 23 **PECEN**: PEC enable

0: PEC calculation disabled

1: PEC calculation enabled

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 28.3: I2C implementation](#).

Bit 22 **ALERTEN**: SMBus alert enable

Device mode (SMBHEN=0):

0: Releases SMBA pin high and Alert Response Address Header disabled: 0001100x followed by NACK.

1: Drives SMBA pin low and Alert Response Address Header enables: 0001100x followed by ACK.

Host mode (SMBHEN=1):

0: SMBus Alert pin (SMBA) not supported.

1: SMBus Alert pin (SMBA) supported.

Note: When ALERTEN=0, the SMBA pin can be used as a standard GPIO.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 28.3: I2C implementation](#).

Bit 21 **SMBDEN**: SMBus Device Default address enable

0: Device default address disabled. Address 0b1100001x is NACKed.

1: Device default address enabled. Address 0b1100001x is ACKed.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 28.3: I2C implementation](#).

Bit 20 **SMBHEN**: SMBus Host address enable

0: Host address disabled. Address 0b0001000x is NACKed.

1: Host address enabled. Address 0b0001000x is ACKed.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 28.3: I2C implementation](#).

Bit 19 **GCEN**: General call enable

0: General call disabled. Address 0b00000000 is NACKed.

1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wakeup from Stop mode enable

0: Wakeup from Stop mode disable.

1: Wakeup from Stop mode enable.

Note: If the Wakeup from Stop mode feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 28.3: I2C implementation](#).

Note: WUPEN can be set only when DNF = '0000'

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.

0: Clock stretching enabled

1: Clock stretching disabled

Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bit 16 **SBC**: Slave byte control

This bit is used to enable hardware byte control in slave mode.

0: Slave byte control disabled

1: Slave byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable

0: DMA mode disabled for reception

1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable

0: DMA mode disabled for transmission

1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF**: Analog noise filter OFF

0: Analog noise filter enabled

1: Analog noise filter disabled

Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter

These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter will filter spikes with a length of up to $DNF[3:0] * t_{I2CCLK}$

0000: Digital filter disabled

0001: Digital filter enabled and filtering capability up to 1 t_{I2CCLK}

...
1111: digital filter enabled and filtering capability up to 15 t_{I2CCLK}

Note: If the analog filter is also enabled, the digital filter is added to the analog filter.

This filter can only be programmed when the I2C is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable

0: Error detection interrupts disabled

1: Error detection interrupts enabled

Note: Any of these errors generate an interrupt:

Arbitration Loss (ARLO)

Bus Error detection (BERR)

Overrun/Underrun (OVR)

Timeout detection (TIMEOUT)

PEC error detection (PECERR)

Alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer Complete interrupt enable

0: Transfer Complete interrupt disabled

1: Transfer Complete interrupt enabled

Note: Any of these events will generate an interrupt:

Transfer Complete (TC)

Transfer Complete Reload (TCR)

Bit 5 **STOPIE**: STOP detection Interrupt enable

0: Stop detection (STOPF) interrupt disabled

1: Stop detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received Interrupt enable

0: Not acknowledge (NACKF) received interrupts disabled

1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match Interrupt enable (slave only)

0: Address match (ADDR) interrupts disabled

1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX Interrupt enable

0: Receive (RXNE) interrupt disabled

1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX Interrupt enable

0: Transmit (TXIS) interrupt disabled

1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable

0: Peripheral disable

1: Peripheral enable

Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.

28.7.2 Control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTO END	RE LOAD	NBYTES[7:0]							
					rs	rw	rw	rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD 10R	ADD10	RD_W RN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw									

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

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address Matched is received, also when PE=0.

0: No PEC transfer.

1: PEC transmission/reception is requested

Note: Writing '0' to this bit has no effect.

This bit has no effect when RELOAD is set.

This bit has no effect in slave mode when SBC=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'.

Please refer to [Section 28.3: I2C implementation](#).

Bit 25 **AUTOEND**: Automatic end mode (master mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in slave mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART will follow).

1: The transfer is not completed after the NBYTES data transfer (NBYTES will be reloaded).

TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC=0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (slave mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address Matched is received, or when PE=0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing '0' to this bit has no effect.

This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: Stop generation (master mode)

The bit is set by software, cleared by hardware when a Stop condition is detected, or when PE = 0.

In Master Mode:

0: No Stop generation.

1: Stop generation after current byte transfer.

Note: Writing '0' to this bit has no effect.

Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing '1' to the ADDRCONF bit in the I2C_ICR register.

0: No Start generation.

1: Restart/Start generation:

- If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer.
- Otherwise setting this bit will generate a START condition once the bus is free.

Note: Writing '0' to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in slave mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)

0: The master sends the complete 10 bit slave address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.

1: The master only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (master mode)

0: The master operates in 7-bit addressing mode,

1: The master operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (master mode)

0: Master requests a write transfer.

1: Master requests a read transfer.

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:8 **SADD[9:8]**: Slave address bit 9:8 (master mode)

In 7-bit addressing mode (ADD10 = 0):

These bits are don't care

In 10-bit addressing mode (ADD10 = 1):

These bits should be written with bits 9:8 of the slave address to be sent

Note: Changing these bits when the START bit is set is not allowed.

Bits 7:1 **SADD[7:1]**: Slave address bit 7:1 (master mode)

In 7-bit addressing mode (ADD10 = 0):

These bits should be written with the 7-bit slave address to be sent

In 10-bit addressing mode (ADD10 = 1):

These bits should be written with bits 7:1 of the slave address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

Bit 0 **SADD0**: Slave address bit 0 (master mode)

In 7-bit addressing mode (ADD10 = 0):

This bit is don't care

In 10-bit addressing mode (ADD10 = 1):

This bit should be written with bit 0 of the slave address to be sent

Note: Changing these bits when the START bit is set is not allowed.

28.7.3 Own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1 MODE	OA1[9:8]		OA1[7:1]							OA1[0]
rw					rw	rw		rw							rw

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

Bit 15 **OA1EN**: Own Address 1 enable

0: Own address 1 disabled. The received slave address OA1 is NACKed.

1: Own address 1 enabled. The received slave address OA1 is ACKed.

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

Bit 10 **OA1MODE** Own Address 1 10-bit mode

0: Own address 1 is a 7-bit address.

1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN=0.

Bits 9:8 **OA1[9:8]**: Interface address

7-bit addressing mode: don't care

10-bit addressing mode: bits 9:8 of address

Note: These bits can be written only when OA1EN=0.

Bits 7:1 **OA1[7:1]**: Interface address

Bits 7:1 of address

Note: These bits can be written only when OA1EN=0.

Bit 0 **OA1[0]**: Interface address

7-bit addressing mode: don't care

10-bit addressing mode: bit 0 of address

Note: This bit can be written only when OA1EN=0.

28.7.4 Own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw			rw							

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

Bit 15 **OA2EN**: Own Address 2 enable

0: Own address 2 disabled. The received slave address OA2 is NACKed.

1: Own address 2 enabled. The received slave address OA2 is ACKed.

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

Bits 10:8 **OA2MSK[2:0]**: Own Address 2 masks

000: No mask

001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.

010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.

011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.

100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.

101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.

110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.

111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN=0.

As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address

bits 7:1 of address

Note: These bits can be written only when OA2EN=0.

Bit 0 Reserved, must be kept at reset value.

28.7.5 Timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait states

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

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale I2CCLK in order to generate the clock period t_{PRESC} used for data setup and hold counters (refer to *I2C timings on page 773*) and for SCL high and low level counters (refer to *I2C master initialization on page 788*).

$$t_{\text{PRESC}} = (\text{PRESC} + 1) \times t_{\text{I2CCLK}}$$

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} between SDA edge and SCL rising edge in transmission mode.

$$t_{\text{SCLDEL}} = (\text{SCLDEL} + 1) \times t_{\text{PRESC}}$$

Note: t_{SCLDEL} is used to generate $t_{\text{SU:DAT}}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge SDA edge in transmission mode.

$$t_{\text{SDADEL}} = \text{SDADEL} \times t_{\text{PRESC}}$$

Note: SDADEL is used to generate $t_{\text{HD:DAT}}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (master mode)

This field is used to generate the SCL high period in master mode.

$$t_{\text{SCLH}} = (\text{SCLH} + 1) \times t_{\text{PRESC}}$$

Note: SCLH is also used to generate $t_{\text{SU:STO}}$ and $t_{\text{HD:STA}}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (master mode)

This field is used to generate the SCL low period in master mode.

$$t_{\text{SCLL}} = (\text{SCLL} + 1) \times t_{\text{PRESC}}$$

Note: SCLL is also used to generate t_{BUF} and $t_{\text{SU:STA}}$ timings.

Note: This register must be configured when the I2C is disabled ($\text{PE} = 0$).

28.7.6 Timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$.

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

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{\text{LOW:EXT}}$ is done by the I2C interface, a timeout error is detected (TIMEOUT=1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

In master mode, the master cumulative clock low extend time ($t_{\text{LOW:MEXT}}$) is detected

In slave mode, the slave cumulative clock low extend time ($t_{\text{LOW:SEXT}}$) is detected

$$t_{\text{LOW:EXT}} = (\text{TIMEOUTB} + 1) \times 2048 \times t_{\text{I2CCLK}}$$

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

$$t_{\text{TIMEOUT}} = (\text{TIMEOUTA} + 1) \times 2048 \times t_{\text{I2CCLK}}$$

– The bus idle condition (both SCL and SDA high) when TIDLE=1

$$t_{\text{IDLE}} = (\text{TIMEOUTA} + 1) \times 4 \times t_{\text{I2CCLK}}$$

Note: These bits can be written only when TIMOUTEN=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Please refer to [Section 28.3: I2C implementation](#).

28.7.7 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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIME OUT	PEC ERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	r_w1	r_w1

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

Bits 23:17 **ADDCODE[6:0]**: Address match code (Slave mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1).

In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the 2 MSBs of the address.

Bit 16 **DIR**: Transfer direction (Slave mode)

This flag is updated when an address match event occurs (ADDR=1).

0: Write transfer, slave enters receiver mode.

1: Read transfer, slave enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a Stop condition is detected, or when PE=0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 28.3: I2C implementation](#).

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 28.3: I2C implementation](#).

Bit 11 PECERR: PEC Error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 28.3: I2C implementation](#).

Bit 10 OVR: Overrun/Underrun (slave mode)

This flag is set by hardware in slave mode with NOSTRETCH=1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 9 ARLO: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 8 BERR: Bus error

This flag is set by hardware when a misplaced Start or Stop condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting BERRCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 7 TCR: Transfer Complete Reload

This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE=0.

This flag is only for master mode, or for slave mode when the SBC bit is set.

Bit 6 TC: Transfer Complete (master mode)

This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE=0.

Bit 5 STOPF: Stop detection flag

This flag is set by hardware when a Stop condition is detected on the bus and the peripheral is involved in this transfer:

- either as a master, provided that the STOP condition is generated by the peripheral.
- or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 4 NACKF: Not Acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 3 ADDR: Address matched (slave mode)

This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting ADDRCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE=0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software when NOSTRETCH=1 only, in order to generate a TXIS event (interrupt if TXIE=1 or DMA request if TXDMAEN=1).

Note: This bit is cleared by hardware when PE=0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE=0.

28.7.8 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	TIM OUTCF	PECCF	OVRCF	ARLO CF	BERR CF	Res.	Res.	STOP CF	NACK CF	ADDR CF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

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

Bit 13 **ALERTCF**: Alert flag clear

Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 28.3: I2C implementation](#).

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 28.3: I2C implementation](#).

Bit 11 **PECCF**: PEC Error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 28.3: I2C implementation](#).

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 ACKF 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.

28.7.9 PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: No wait states

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

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

Bits 7:0 **PEC[7:0]** Packet error checking register

This field contains the internal PEC when PECEN=1.

The PEC is cleared by hardware when PE=0.

Note: *If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Please refer to [Section 28.3: I2C implementation](#).*

28.7.10 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							

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

Bits 7:0 **RXDATA[7:0]** 8-bit receive data

Data byte received from the I²C bus.

28.7.11 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							

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

Bits 7:0 **TXDATA[7:0]** 8-bit transmit data

Data byte to be transmitted to the I²C bus.

Note: These bits can be written only when TXE=1.

28.7.12 I2C register map

The table below provides the I2C register map and reset values.

Table 151. I2C register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x0	I2C_CR1	Res	Res	Res	Res	Res	Res	Res	Res	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res	ANFOFF	DNF[3:0]			ERRIE			TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE		
	Reset value									0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0			
0x4	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			
0x8	I2C_OAR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OA1EN	Res	Res	Res	Res	OA1MODE	OA1[9:0]												
	Reset value																	0	Res	Res	Res		0	0	0	0	0	0	0	0	0	0				
0xC	I2C_OAR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OA2EN	Res	Res	Res	Res	OA2MSK [2:0]	OA2[7:1]					Res							
	Reset value																	0		Res	Res		0	0	0	0	0	0	0	0	0					
0x10	I2C_TIMINGR	PRESC[3:0]				Res	Res	Res	Res	SCLDEL[3:0]				SDADEL[3:0]			SCLH[7:0]							SCLL[7:0]												
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x14	I2C_TIMEOUTR	TEXTEN	Res	Res	Res	Res	TIMEOUTB[11:0]													TIMOUTEN	Res	TITLE	TIMEOUTA[11:0]													
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	Res		0	0	0	0	0	0	0	0	0	0	0	0			
0x18	I2C_ISR	Res	Res	Res	Res	Res	Res	Res	Res	ADDCODE[6:0]							DIR	BUSY	Res	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXF			
	Reset value									0	0	0	0	0	0	0	0	0	0	Res	0	0	0	0	0	0	0	0	0	0	0	0	1			
0x1C	I2C_ICR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ALERTCF	TIMEOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res	Res	STOPCF	NACKCF	ADDRCF	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	PEC[7:0]										
	Reset value																									0	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	RXDATA[7:0]										
	Reset value																									0	0	0	0	0	0	0	0			

Table 151. I2C register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
0x28	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.	TXDATA[7:0]													
	Reset value																									0	0	0	0	0	0	0	0					

Refer to [Section 3.2.2 on page 48](#) for the register boundary addresses.

29 Universal synchronous asynchronous receiver transmitter (USART)

29.1 Introduction

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The USART offers a very wide range of baud rates using a fractional baud rate generator.

It supports synchronous one-way communication and half-duplex single wire communication, as well as multiprocessor communications. It also supports the LIN (Local Interconnect Network), smartcard protocol and IrDA (Infrared Data Association) SIR ENDEC specifications and modem operations (CTS/RTS).

High speed data communication is possible by using the DMA (direct memory access) for multibuffer configuration.

29.2 USART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to give flexibility between speed and clock tolerance
- A common programmable transmit and receive baud rate of up to 9 Mbit/s when the clock frequency is 72 MHz and oversampling is by 8
- Dual clock domain allowing:
 - USART functionality and wakeup from Stop mode
 - Convenient baud rate programming independent from the PCLK reprogramming
- 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 mode and clock output for synchronous communications
- 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
- Fourteen interrupt sources with flags
- Multiprocessor communications

The USART enters mute mode if the address does not match.
- Wakeup from mute mode (by idle line detection or address mark detection)

29.3 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
 - 1.5 stop bits for smartcard operation
- Support for ModBus communication
 - Timeout feature
 - CR/LF character recognition

29.4 USART implementation

Table 152. STM32F302xx USART features

USART modes/features ⁽¹⁾	STM32F302xB/C		STM32F302xD/E		STM32F302x6/8	
	USART1/ USART2/ USART3	USART4/ USART5	USART1/ USART2/ USART3	USART4/ USART5	USART1	USART2/ USART3
Hardware flow control for modem	X	-	X	-	X	X
Continuous communication using DMA	X	X	X	X	X	X
Multiprocessor communication	X	X	X	X	X	X
Synchronous mode	X	-	X	-	X	X
Smartcard mode	X ⁽²⁾⁽³⁾	-	X ⁽⁴⁾	-	X ⁽⁴⁾	-
Single-wire half-duplex communication	X	X	X	X	X	X
IrDA SIR ENDEC block	X	X	X	X	X	-
LIN mode	X	X	X	X	X	-
Dual clock domain and wakeup from Stop mode	X	X	X	X	X	-
Receiver timeout interrupt	X	X	X	X	X	-
Modbus communication	X	X	X	X	X	-
Auto baud rate detection	X (4 modes)	-	X (4 modes)	-	X (4 modes)	-
Driver Enable	X	-	X	-	X	X
USART data length	8 and 9 bits		7, 8 and 9 bits		7, 8 and 9 bits	

1. X = supported.

2. SCLK output is disabled when UE bit = 0.

3. With the following limitation for STM32F302xB/C: If the USART is used in smartcard mode and the card cannot use the default communication parameters after Answer To Reset and doesn't support clock stop, it is not possible to use SCLK to clock the card. This is due to the fact that the USART and its clock output must be disabled while reprogramming some of the parameters.

4. SCLK is always available when CLKEN = 1, regardless of the UE bit value.

29.5 USART functional description

Any USART bidirectional communication requires a minimum of two pins: Receive data In (RX) and Transmit data Out (TX):

- **RX:** Receive data Input.

This is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.

- **TX:** Transmit data Output.

When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire and Smartcard modes, this I/O is used to transmit and receive the data.

Serial data are transmitted and received through these pins in normal USART mode. The frames are comprised of:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (7, 8 or 9 bits) least significant bit first
- 1, 1.5, 2 stop bits indicating that the frame is complete
- The USART interface uses a baud rate generator
- A status register (USARTx_ISR)
- Receive and transmit data registers (USARTx_RDR, USARTx_TDR)
- A baud rate register (USARTx_BRR)
- A guard-time register (USARTx_GTPR) in case of Smartcard mode.

Refer to [Section 29.8: USART registers on page 879](#) for the definitions of each bit.

The following pin is required to interface in synchronous mode and Smartcard mode:

- **SCLK:** Clock output. This pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX. This can be used to control peripherals that have shift registers. The clock phase and polarity are software programmable. In Smartcard mode, SCLK output can provide the clock to the smartcard.

The following pins are required in RS232 Hardware flow control mode:

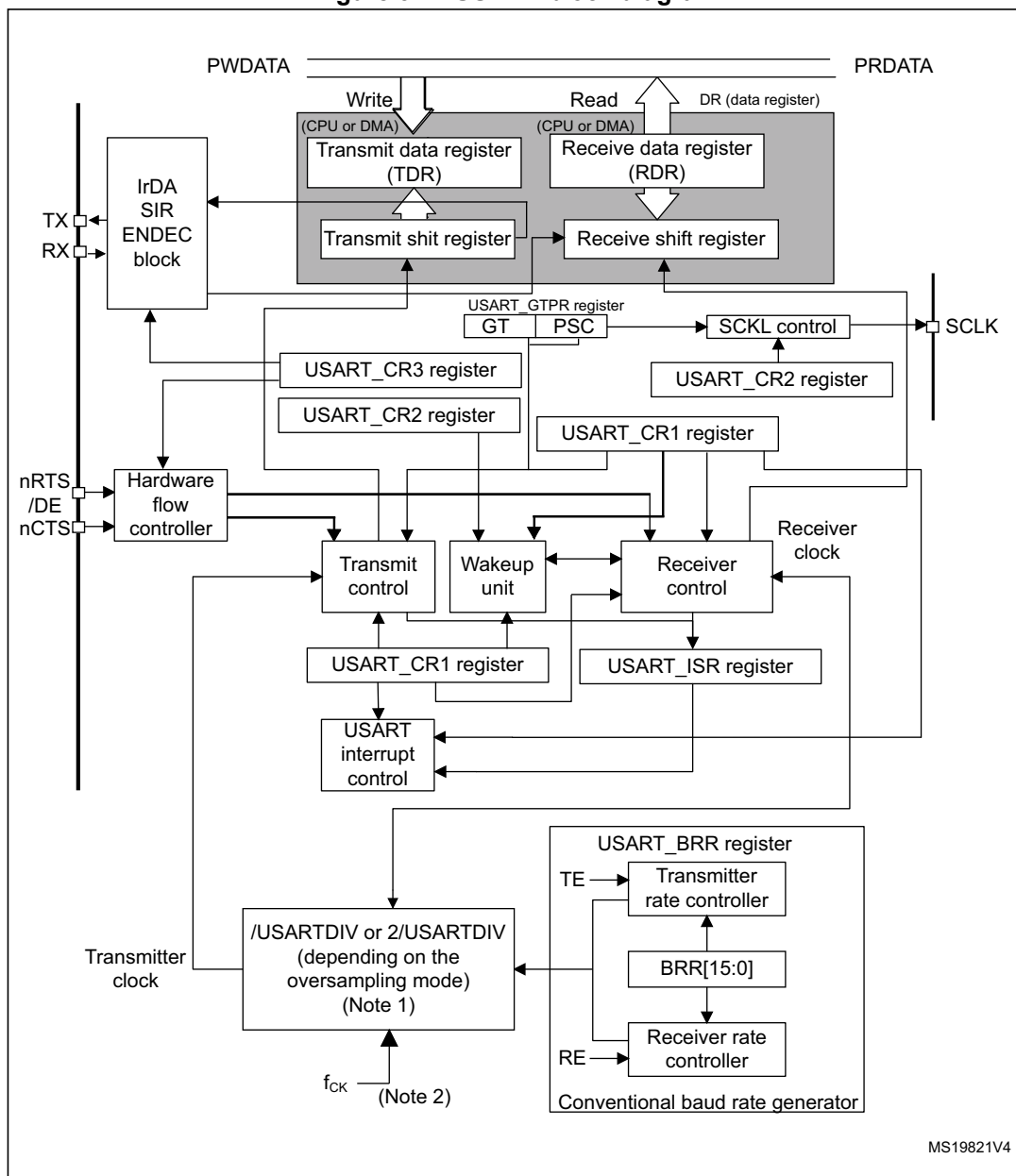
- **nCTS:** Clear To Send blocks the data transmission at the end of the current transfer when high
- **nRTS:** Request to send indicates that the USART is ready to receive data (when low).

The following pin is required in RS485 Hardware control mode:

- **DE:** Driver Enable activates the transmission mode of the external transceiver.

Note: **DE** and **nRTS** share the same pin.

Figure 314. USART block diagram



1. For details on coding USARTDIV in the USARTx_BRR register, please refer to [Section 29.5.4: Baud rate generation](#).
2. f_{CK} can be f_{LSE} , f_{HSI} , f_{PCLK} , f_{SYS} .

29.5.1 USART character description

The word length can be selected as being either 7 or 8 or 9 bits by programming the M[1:0] bits in the USARTx_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 Autobaudrate (0x7F and 0x55 frames detection) are not supported. 7-bit mode is supported only on some USARTs.

In default configuration, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

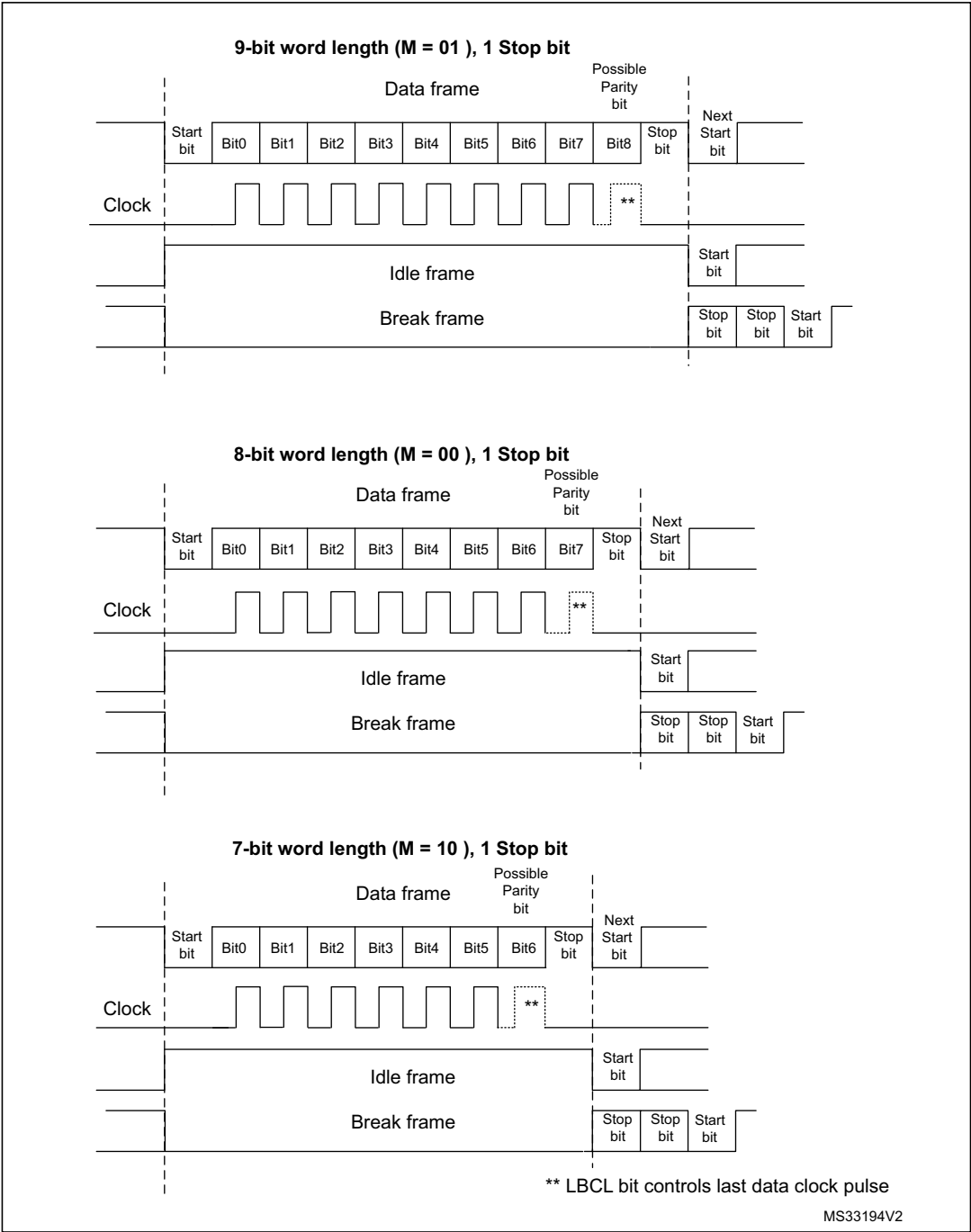
An **Idle character** is interpreted as an entire frame of “1”s. (The number of “1” ‘s will include the number of stop bits).

A **Break character** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.

The details of each block is given below.

Figure 315. Word length programming



29.5.2 Transmitter

The transmitter can send data words of either 7, 8 or 9 bits depending on the M bits status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin and the corresponding clock pulses are output on the SCLK pin.

Character transmission

During an USART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the USARTx_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see [Figure 314](#)).

Every character is preceded by a start bit which is a logic level low for one bit period. The character is terminated by a configurable number of stop bits.

The following stop bits are supported by USART: 1, 1.5 and 2 stop bits.

Note:

The TE bit must be set before writing the data to be transmitted to the USARTx_TDR.

The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission will corrupt the data on the TX pin as the baud rate counters will get frozen. The current data being transmitted will be lost.

An idle frame will be sent after the TE bit is enabled.

Configurable stop bits

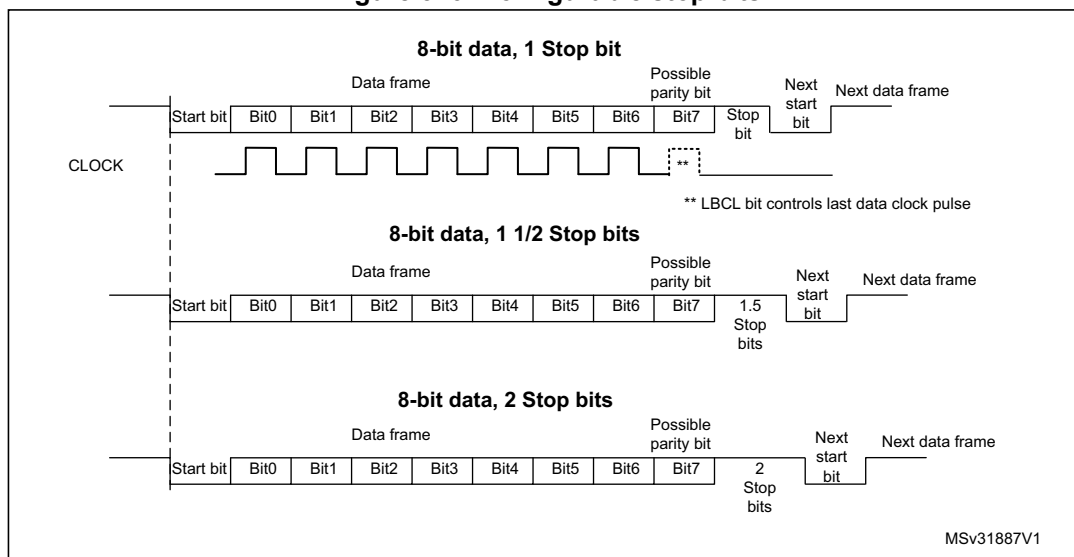
The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.

- **1 stop bit:** This is the default value of number of stop bits.
- **2 stop bits:** This will be supported by normal USART, single-wire and modem modes.
- **1.5 stop bits:** To be used in Smartcard mode.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits (when M[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

1. Program the M bits in USARTx_CR1 to define the word length.
2. Select the desired baud rate using the USARTx_BRR register.
3. Program the number of stop bits in USARTx_CR2.
4. Enable the USART by writing the UE bit in USARTx_CR1 register to 1.
5. Select DMA enable (DMAT) in USARTx_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the TE bit in USARTx_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USARTx_TDR register (this clears the TXE bit). Repeat this for each data to be transmitted in case of single buffer.
8. After writing the last data into the USARTx_TDR register, wait until TC=1. This indicates that the transmission of the last frame is complete. This is required for instance when the USART is disabled or enters the Halt mode to avoid corrupting the last transmission.

Single byte communication

Clearing the TXE bit is always performed by a write to the transmit data register.

The TXE bit is set by hardware and it indicates:

- The data has been moved from the USARTx_TDR register to the shift register and the data transmission has started.
- The USARTx_TDR register is empty.
- The next data can be written in the USARTx_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

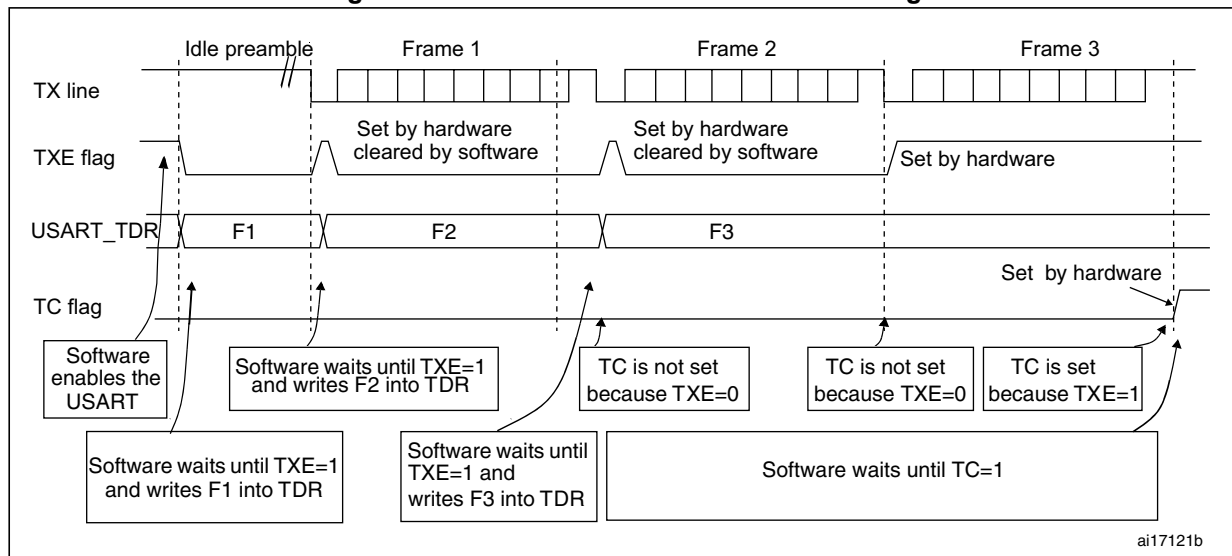
When a transmission is taking place, a write instruction to the USARTx_TDR register stores the data in the TDR register; next, the data is copied in the shift register at the end of the currently ongoing transmission.

When no transmission is taking place, a write instruction to the USARTx_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

If a frame is transmitted (after the stop bit) and the TXE bit is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the USARTx_CR1 register.

After writing the last data in the USARTx_TDR register, it is mandatory to wait for TC=1 before disabling the USART or causing the microcontroller to enter the low-power mode (see [Figure 317: TC/TXE behavior when transmitting](#)).

Figure 317. TC/TXE behavior when transmitting



Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see [Figure 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.

In the case 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.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

29.5.3 Receiver

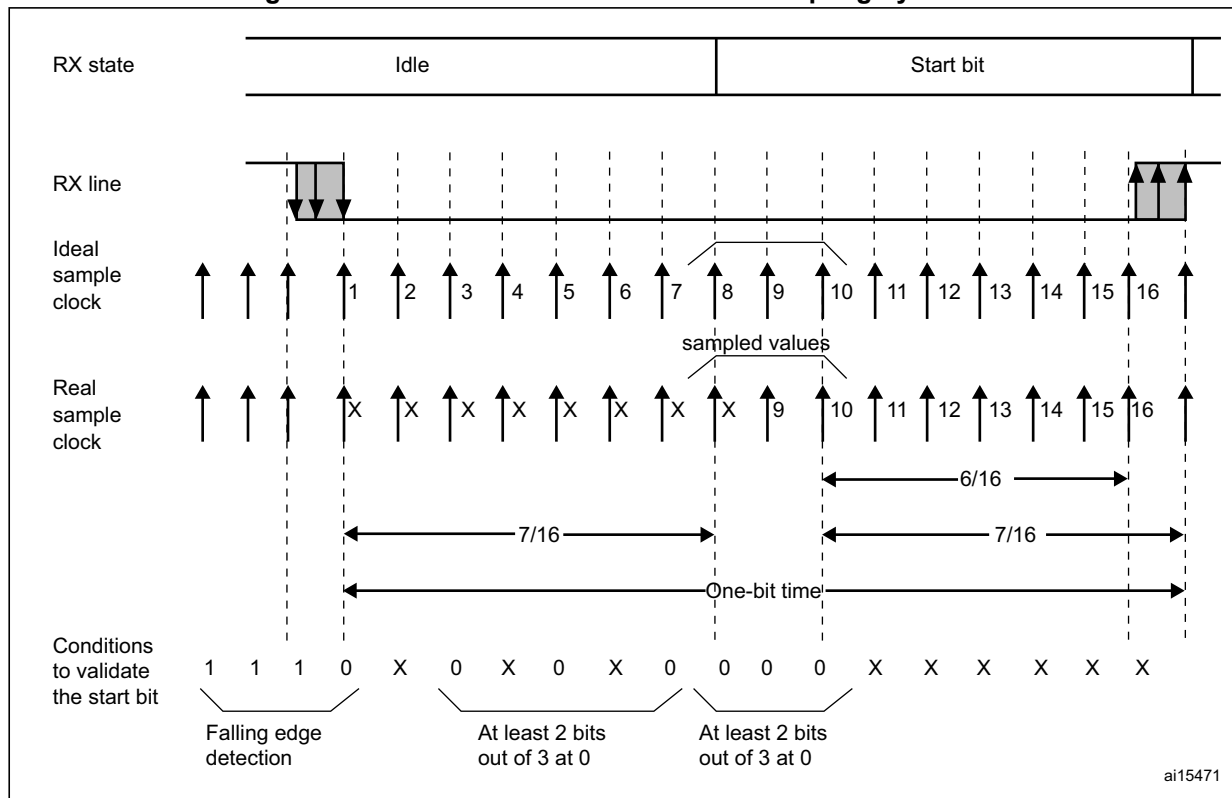
The USART can receive data words of either 7, 8 or 9 bits depending on the M bits in the USARTx_CR1 register.

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, interrupt generated if RXNEIE=1) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated (RXNE flag set, interrupt generated if RXNEIE=1) but the NF noise flag is set if,

a. for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b. for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither conditions a. or b. are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data shifts in least significant bit first (default configuration) through the RX pin. In this mode, the USARTx_RDR register consists of a buffer (RDR) between the internal bus and the receive shift register.

Character reception procedure

1. Program the M bits in USARTx_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USARTx_BRR
3. Program the number of stop bits in USARTx_CR2.
4. Enable the USART by writing the UE bit in USARTx_CR1 register to 1.
5. Select DMA enable (DMAR) in USARTx_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the RE bit USARTx_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- The RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- An interrupt is generated if the RXNEIE bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception. PE flag can also be set with RXNE.
- In multibuffer, RXNE is set after every byte received and is cleared by the DMA read of the Receive data Register.
- In single buffer mode, clearing the RXNE bit is performed by a software read to the USARTx_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USARTx_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, there is the same procedure as for a received data character plus an interrupt if the IDLEIE bit is set.

Overrun error

An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared.

The RXNE flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

- The ORE bit is set.
- The RDR content will not be lost. The previous data is available when a read to USARTx_RDR is performed.
- The shift register will be overwritten. After that point, any data received during overrun is lost.
- An interrupt is generated if either the RXNEIE bit is set or EIE bit is set.
- The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. There are two possibilities:

- if RXNE=1, then the last valid data is stored in the receive register RDR and can be read,
- if RXNE=0, then it means that the last valid data has already been read and thus there is nothing to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.

Selecting the clock source and the proper oversampling method

The choice of the clock source is done through the Clock Control system (see Section Reset and clock control (RCC)). The clock source must be chosen before enabling the USART (by setting the UE bit).

The choice of the clock source must be done according to two criteria:

- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is f_{CK} .

When the dual clock domain with the wakeup from Stop mode is supported, the clock source can be one of the following sources: PCLK (default), LSE, HSI or SYSCLK. Otherwise, the USART clock source is PCLK.

Choosing LSE or HSI as clock source may allow the USART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selection, the USART wakes up the MCU, when needed, in order to transfer the received data by software reading the USARTx_RDR register or by DMA.

For the other clock sources, the system must be active in order to allow USART communication.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This allows a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USARTx_CR1 register and can be either 16 or 8 times the baud rate clock ([Figure 319](#) and [Figure 320](#)).

Depending on the application:

- Select oversampling by 8 (OVER8=1) to achieve higher speed (up to $f_{CK}/8$). In this case the maximum receiver tolerance to clock deviation is reduced (refer to [Section 29.5.5: Tolerance of the USART receiver to clock deviation on page 854](#))
- Select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum $f_{CK}/16$ where f_{CK} is the clock source frequency.

Programming the ONEBIT bit in the USARTx_CR3 register selects the method used to evaluate the logic level. There are two options:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NF bit is set
- A single sample in the center of the received bit

Depending on the application:

- select the three samples' majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to [Figure 153](#)) because this indicates that a glitch occurred during the sampling.
- select the single sample method (ONEBIT=1) when the line is noise-free to increase the receiver's tolerance to clock deviations (see [Section 29.5.5: Tolerance of the USART receiver to clock deviation on page 854](#)). In this case the NF bit will never be set.

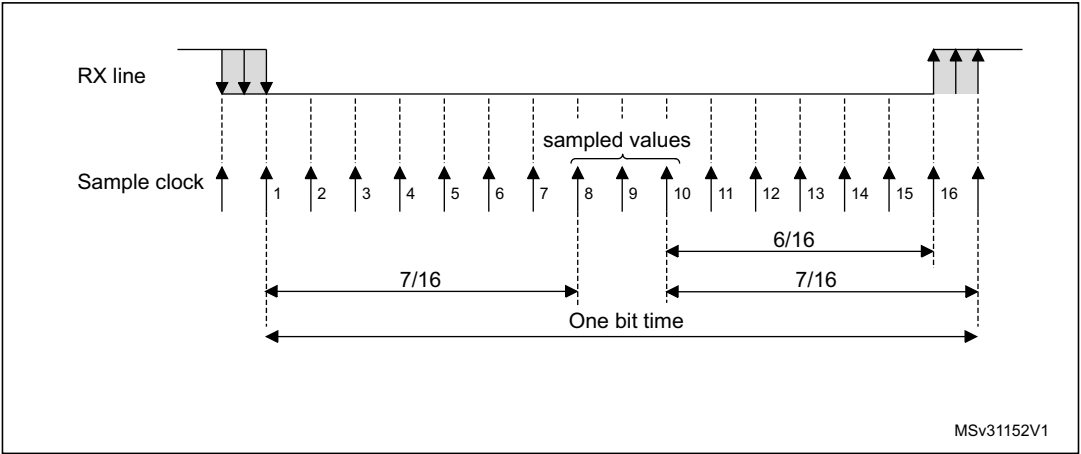
When noise is detected in a frame:

- The NF bit is set at the rising edge of the RXNE bit.
- The invalid data is transferred from the Shift register to the USARTx_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USARTx_CR3 register.

The NF bit is reset by setting NCF bit in ICR register.

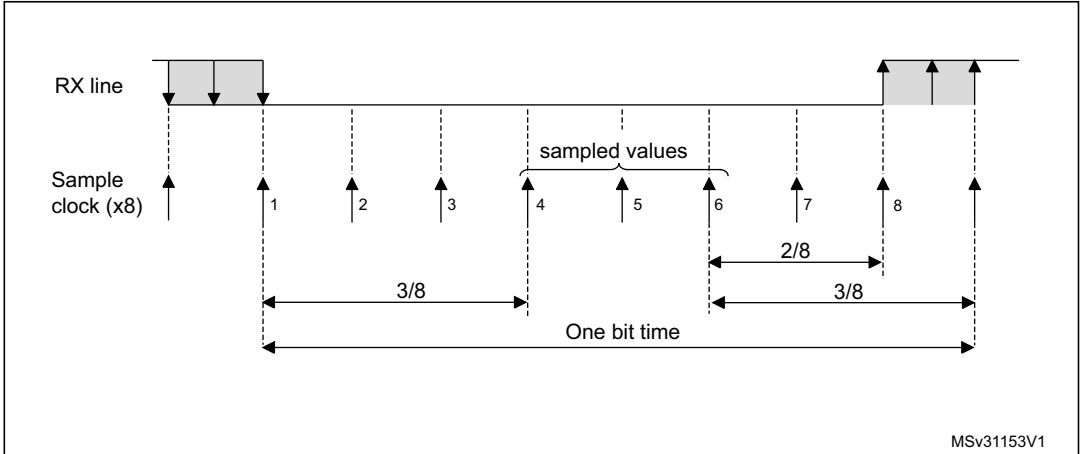
Note: *Oversampling by 8 is not available in LIN, smartcard and IrDA modes. In those modes, the OVER8 bit is forced to '0' by hardware.*

Figure 319. Data sampling when oversampling by 16



MSv31152V1

Figure 320. Data sampling when oversampling by 8



MSv31153V1

Table 153. 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 USARTx_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USARTx_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USARTx_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of Control Register 2 - it can be either 1 or 2 in normal mode and 1.5 in Smartcard mode.

- **1 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 is correctly sent. Thus the receiver block must be enabled (RE = 1 in the USARTx_CR1 register) and the stop bit is checked to test if the smartcard has detected a parity error. In the event of a parity error, the smartcard forces the data signal low during the sampling - NACK signal-, which is flagged as a framing error. Then, the FE flag is set with the RXNE at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be decomposed into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through. Refer to [Section 29.5.13: Smartcard mode on page 865](#) 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. If a framing error is detected during the first stop bit the framing error flag will be set. The second stop bit is not checked for framing error. The RXNE flag will be set at the end of the first stop bit.

29.5.4 Baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the USARTx_BRR register.

Equation 1: Baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the equation is:

$$\text{Tx/Rx baud} = \frac{f_{\text{CK}}}{\text{USARTDIV}}$$

In case of oversampling by 8, the equation is:

$$\text{Tx/Rx baud} = \frac{2 \times f_{\text{CK}}}{\text{USARTDIV}}$$

Equation 2: Baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

In smartcard, LIN and IrDA modes, only Oversampling by 16 is supported:

$$\text{Tx/Rx baud} = \frac{f_{\text{CK}}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USARTx_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 USARTx_BRR. Hence the baud rate register value should not be changed during communication.

In case of oversampling by 16 or 8, USARTDIV must be greater than or equal to 16d.

How to derive USARTDIV from USARTx_BRR register values

Example 1

To obtain 9600 baud with $f_{\text{CK}} = 8 \text{ MHz}$.

- In case of oversampling by 16:
 - USARTDIV = $8\,000\,000/9600$
 - BRR = USARTDIV = 833d = 0341h
- In case of oversampling by 8:
 - USARTDIV = $2 * 8\,000\,000/9600$
 - USARTDIV = 1666,66 (1667d = 683h)
 - BRR[3:0] = 3h >> 1 = 1h
 - BRR = 0x681

Example 2

To obtain 921.6 Kbaud with $f_{CK} = 48$ MHz.

- In case of oversampling by 16:
 $USARTDIV = 48\,000\,000/921\,600$
 $BRR = USARTDIV = 52d = 34h$
- In case of oversampling by 8:
 $USARTDIV = 2 * 48\,000\,000/921\,600$
 $USARTDIV = 104$ ($104d = 68h$)
 $BRR[3:0] = USARTDIV[3:0] >> 1 = 8h >> 1 = 4h$
 $BRR = 0x64$

Table 154. Error calculation for programmed baud rates at $f_{CK} = 72$ MHz in both cases of oversampling by 16 or by 8⁽¹⁾

Baud rate		Oversampling by 16 (OVER8 = 0)			Oversampling by 8 (OVER8 = 1)		
S.No	Desired	Actual	BRR	% Error = (Calculated - Desired)B.Rate/ Desired B.Rate	Actual	BRR	% Error
1	2.4 KBps	2.4 KBps	0x7530	0	2.4 KBps	0xEA60	0
2	9.6 KBps	9.6 KBps	0x1D4C	0	9.6 KBps	0x3A94	0
3	19.2 KBps	19.2 KBps	0xEA6	0	19.2 KBps	0x1D46	0
4	38.4 KBps	38.4 KBps	0x753	0	38.4 KBps	0xEA3	0
5	57.6 KBps	57.6 KBps	0x4E2	0	57.6 KBps	0x9C2	0
6	115.2 KBps	115.2 KBps	0x271	0	115.2 KBps	0x4E1	0
7	230.4 KBps	230.03KBps	0x139	0.16	230.4 KBps	0x270	0
8	460.8 KBps	461.54KBps	0x9C	0.16	460.06KBps	0x134	0.16
9	921.6 KBps	923.08KBps	0x4E	0.16	923.07KBps	0x96	0.16
10	2 MBps	2 MBps	0x24	0	2 MBps	0x44	0
11	3 MBps	3 MBps	0x18	0	3 MBps	0x30	0
12	4MBps	4MBps	0x12	0	4MBps	0x22	0
13	5MBps	N.A	N.A	N.A	4965.51KBps	0x16	0.69
14	6MBps	N.A	N.A	N.A	6MBps	0x14	0
15	7MBps	N.A	N.A	N.A	6857.14KBps	0x12	2
16	9MBps	N.A	N.A	N.A	9MBps	0x10	0

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

29.5.5 Tolerance of the USART receiver to clock deviation

The asynchronous receiver of the USART works correctly only if the total clock system deviation is less than the tolerance of the USART receiver. The causes which contribute to the total deviation are:

- DTRA: Deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: Error due to the baud rate quantization of the receiver
- DREC: Deviation of the receiver's local oscillator
- DTCL: Deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{USART receiver's tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from Stop mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUSTOP}}{11 \times T_{bit}}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUSTOP}}{10 \times T_{bit}}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUSTOP}}{9 \times T_{bit}}$$

t_{WUSTOP} is the wakeup time from Stop mode, which is specified in the product datasheet.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in [Table 155](#) and [Table 156](#) depending on the following choices:

- 9-, 10- or 11-bit character length defined by the M bits in the USARTx_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USARTx_CR1 register
- Bits BRR[3:0] of USARTx_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 USARTx_CR3 register.

Table 155. 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 156. 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 155](#) and [Table 156](#) may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit durations when M bits = 00 (11-bit durations when M bits = 01 or 9-bit durations when M bits = 10).

29.5.6 Auto baud rate detection

The USART is able to detect and automatically set the USARTx_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance
- The system is using a relatively low accuracy clock source and this mechanism allows the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed (when oversampling by 16, the baud rate is between $f_{CK}/65535$ and $f_{CK}/16$. when oversampling by 8, the baudrate is between $f_{CK}/65535$ and $f_{CK}/8$).

Before activating the auto baud rate detection, the auto baud rate detection mode must be chosen. There are various modes based on different character patterns.

They can be chosen through the ABRMOD[1:0] field in the USARTx_CR2 register. 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:

- **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, ensuring 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 baudrate is updated first at the end of the

start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit 0 to bit 6 are sampled at BRs while further bits of the character are sampled at BR6.

- **Mode 3:** A 0x55 character frame. In this case, the baudrate is updated first at the end of the start bit (BRs), then at the end of bit 0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit 6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6.

In parallel, another check is performed for each intermediate transition of RX line. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating auto baud rate detection, the USARTx_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 USARTx_CR2 register. The USART will then wait for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USARTx_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag will be set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The RXNE interrupt will signal the end of the operation.

At any later time, the auto baud rate detection may be relaunched by resetting the ABRF flag (by writing a 0).

Note: If the USART is disabled (UE=0) during an auto baud rate operation, the BRR value may be corrupted.

29.5.7 Multiprocessor communication

In multiprocessor communication, the following bits are to be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL, IREN and SCEN bits in the USART_CR3 register.

It is possible to perform multiprocessor communication with the USART (with several USARTs connected in a network). For instance one of the USARTs can be the master, its TX output connected to the RX inputs of the other USARTs. The others are slaves, their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non addressed devices may be placed in mute mode by means of the muting function. In order to use the mute mode feature, the MME bit must be set in the USARTx_CR1 register.

In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in USARTx_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USARTx_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 USARTx_CR1 register:

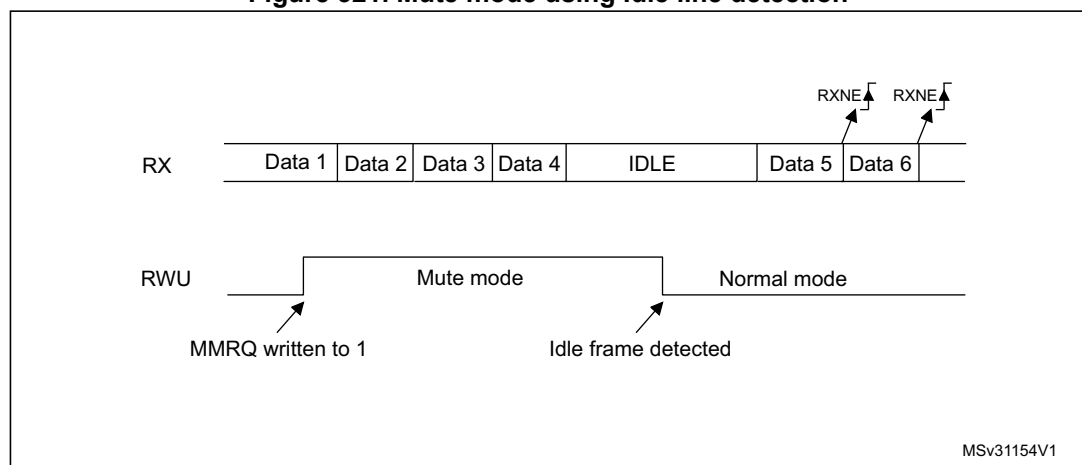
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the USARTx_ISR register. An example of mute mode behavior using Idle line detection is given in [Figure 321](#).

Figure 321. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode will not be entered (RWU is not set).

If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a '1' otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4-bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USARTx_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

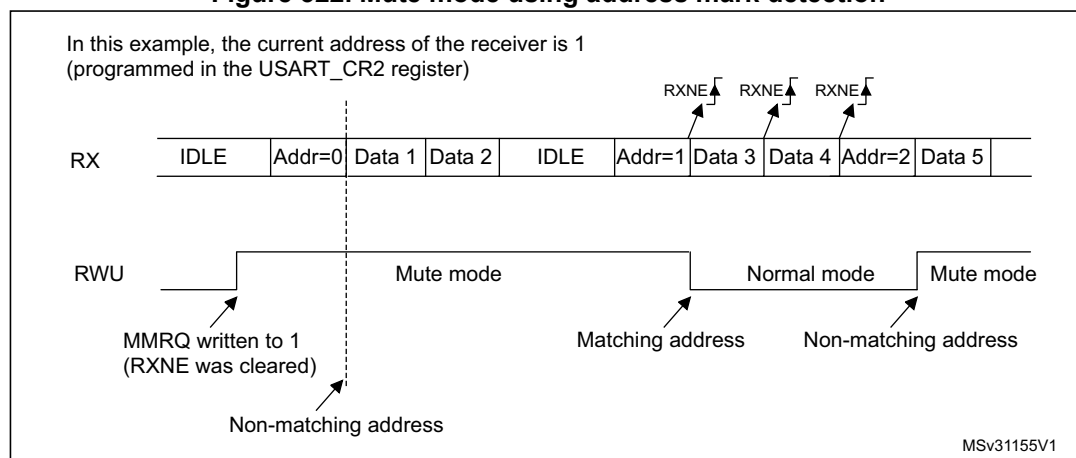
The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode.

The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE bit is set for the address character since the RWU bit has been cleared.

An example of mute mode behavior using address mark detection is given in [Figure 322](#).

Figure 322. Mute mode using address mark detection



29.5.8 Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half duplex, block transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USARTx_CR2 register and the RTOIE in the USARTx_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit duration) 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.

29.5.9 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USARTx_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in [Table 157](#).

Table 157. Frame formats

M bits	PCE bit	USART frame ⁽¹⁾
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 7-bit 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 (9th, 8th or 7th, depending on the M bits 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 bits values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit will be 0 if even parity is selected (PS bit in USARTx_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 bits values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit will be 1 if odd parity is selected (PS bit in USARTx_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USARTx_ISR register and an interrupt is generated if PEIE is set in the USARTx_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USARTx_ICR register.

Parity generation in transmission

If the PCE bit is set in USARTx_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)).

29.5.10 LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Please refer to [Section 29.4: USART implementation on page 838](#).

The LIN mode is selected by setting the LINEN bit in the USARTx_CR2 register. In LIN mode, the following bits must be kept cleared:

- CLKEN in the USARTx_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USARTx_CR3 register.

LIN transmission

The procedure explained in [Section 29.5.2: Transmitter](#) 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 bits to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 ‘0’ bits as a break character. Then 2 bits of value ‘1’ are sent to allow the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in USARTx_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 USARTx_CR2) or 11 (when LBDL=1 in USARTx_CR2) consecutive bits are detected as ‘0’, and are followed by a delimiter character, the LBDF flag is set in USARTx_ISR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1’ is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at ‘0’, which will be the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1’, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the [Figure 323: Break detection in LIN mode \(11-bit break length - LBDL bit is set\) on page 861](#).

Examples of break frames are given on [Figure 324: Break detection in LIN mode vs. Framing error detection on page 862](#).

Figure 323. Break detection in LIN mode (11-bit break length - LBDL bit is set)

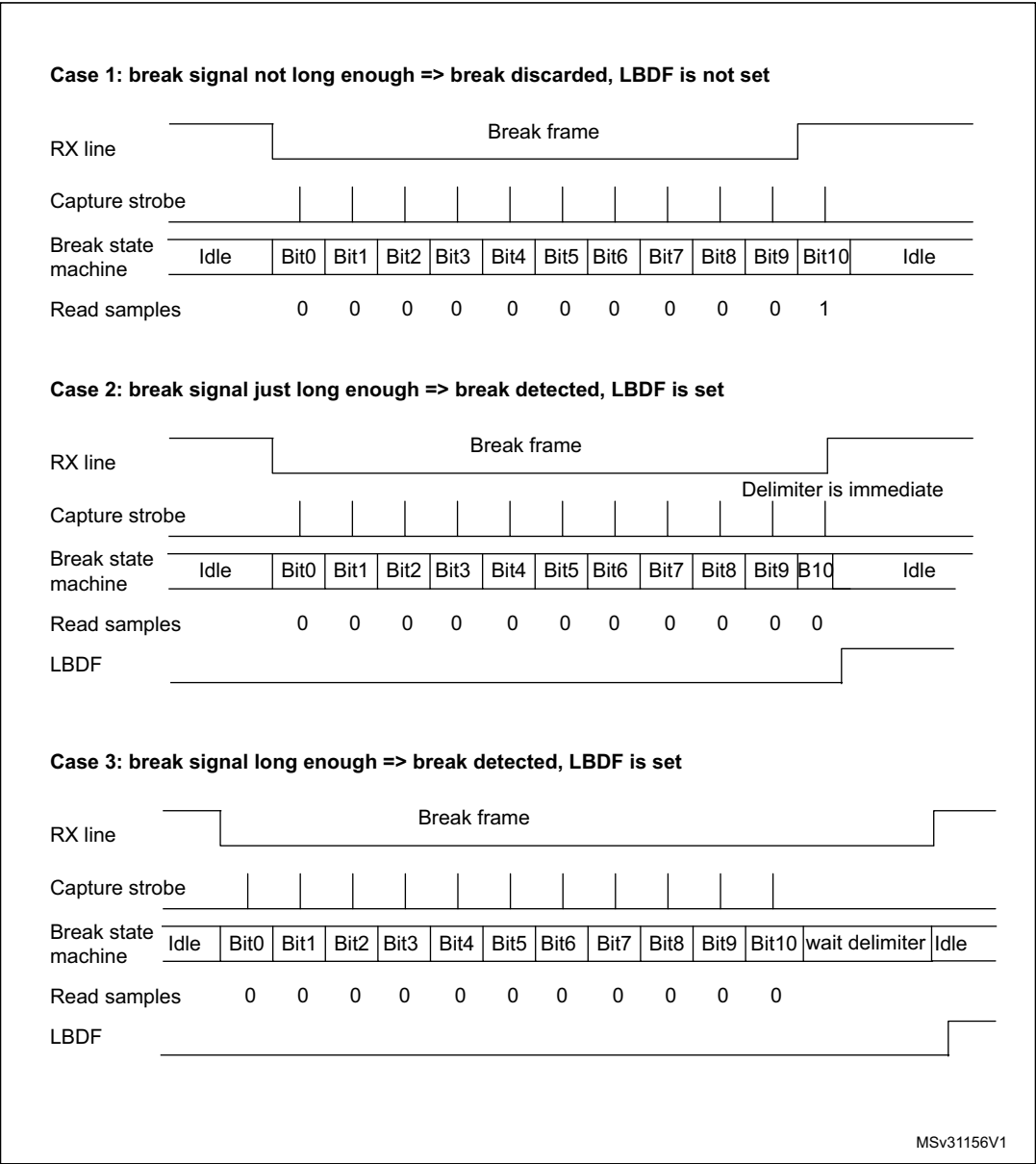
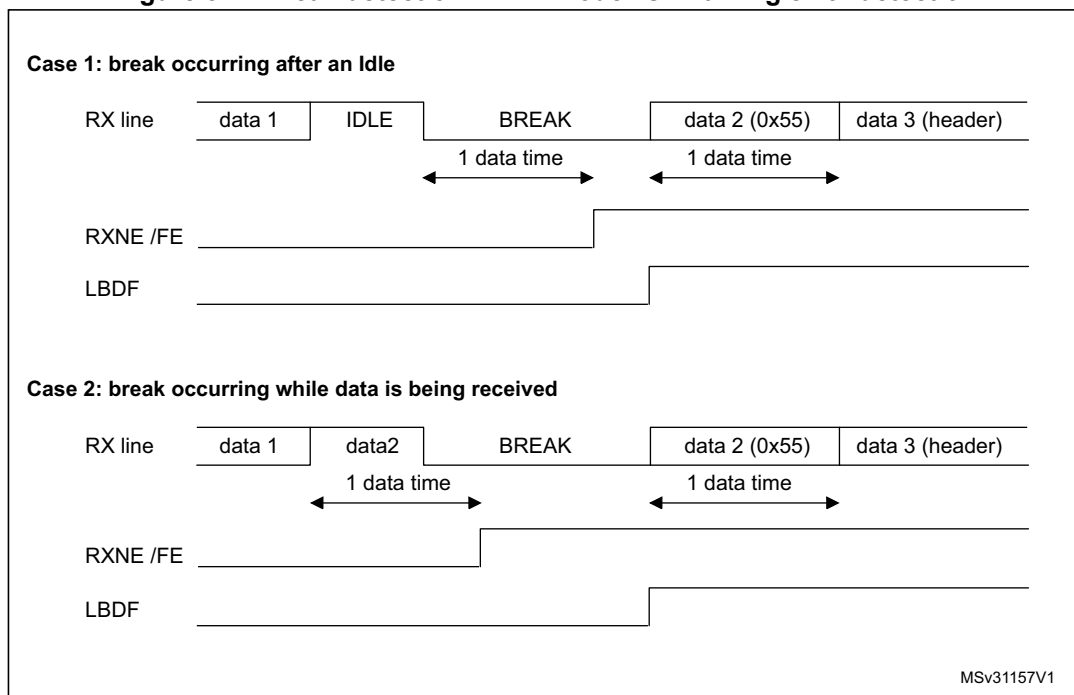


Figure 324. Break detection in LIN mode vs. Framing error detection



29.5.11 USART synchronous mode

The synchronous mode is selected by writing the CLKEN bit in the USARTx_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USARTx_CR2 register,
- SCEN, HDSEL and IREN bits in the USARTx_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USARTx_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USARTx_CR2 register is used to select the clock polarity, and the CPHA bit in the USARTx_CR2 register is used to select the phase of the external clock (see [Figure 325](#), [Figure 326](#) and [Figure 327](#)).

During the Idle state, preamble and send break, the external SCLK clock is not activated.

In synchronous mode the USART transmitter works exactly like in asynchronous mode. But as SCLK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In this mode the USART receiver works in a different manner compared to the asynchronous mode. If RE=1, the data is sampled on SCLK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A setup and a hold time must be respected (which depends on the baud rate: 1/16 bit duration).

Note: *The SCLK pin works in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and data is being transmitted (the data register USARTx_TDR*

written). This means that it is not possible to receive synchronous data without transmitting data.

The LBCL, CPOL and CPHA bits have to be selected when the USART is disabled (UE=0) to ensure that the clock pulses function correctly.

Figure 325. USART example of synchronous transmission

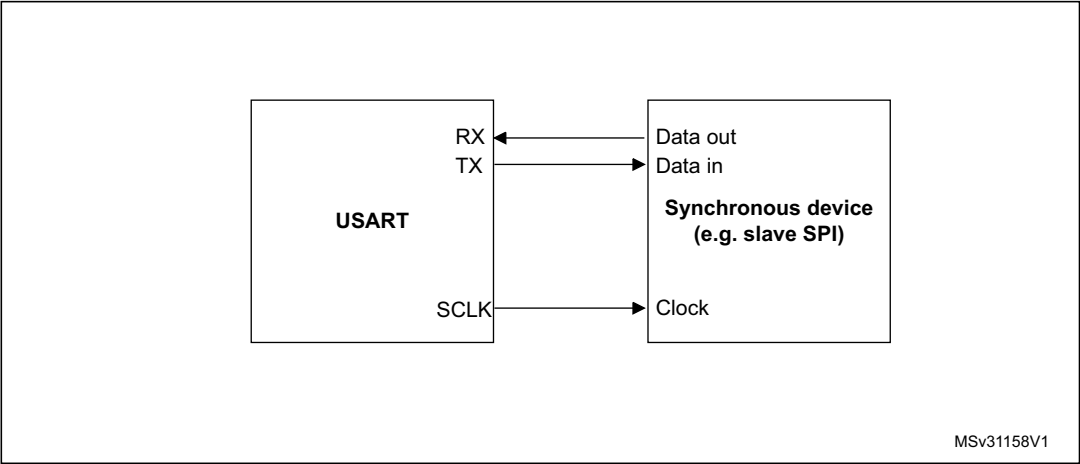


Figure 326. USART data clock timing diagram (M bits = 00)

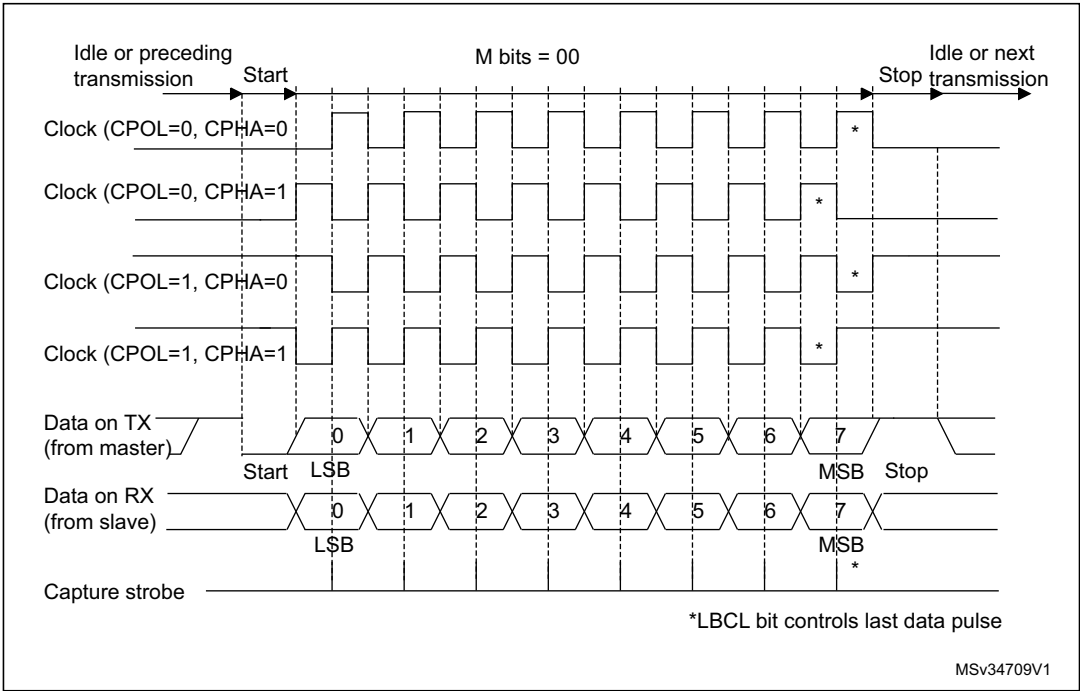


Figure 327. USART data clock timing diagram (M bits = 01)

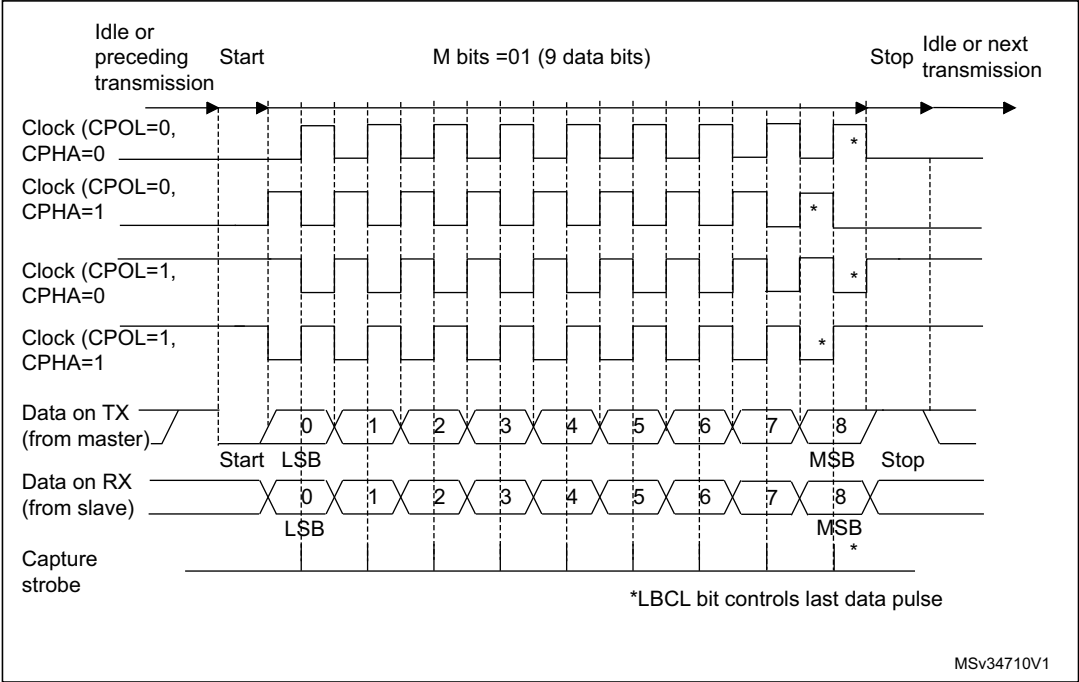
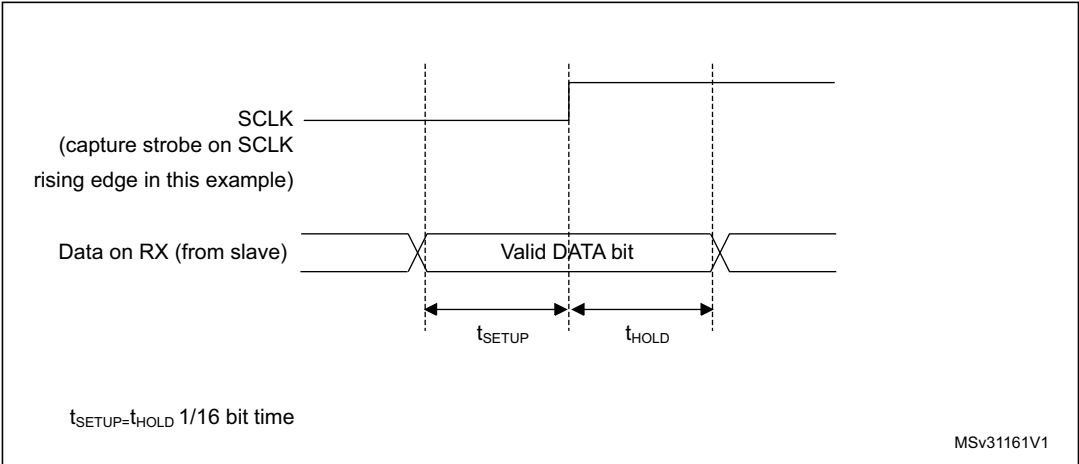


Figure 328. RX data setup/hold time



Note: The function of SCLK is different in Smartcard mode. Refer to [Section 29.5.13: Smartcard mode](#) for more details.

29.5.12 Single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USARTx_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USARTx_CR2 register,
- SCEN and IREN bits in the USARTx_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 USARTx_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflicts on the line must be managed by software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

29.5.13 Smartcard mode

This section is relevant only when Smartcard mode is supported. Please refer to [Section 29.4: USART implementation on page 838](#).

Smartcard mode is selected by setting the SCEN bit in the USARTx_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USARTx_CR2 register,
- HDSEL and IREN bits in the USARTx_CR3 register.

Moreover, the CLKEN bit may be set in order to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous protocol for smartcards as defined in the ISO 7816-3 standard. Both T=0 (character mode) and T=1 (block mode) are supported.

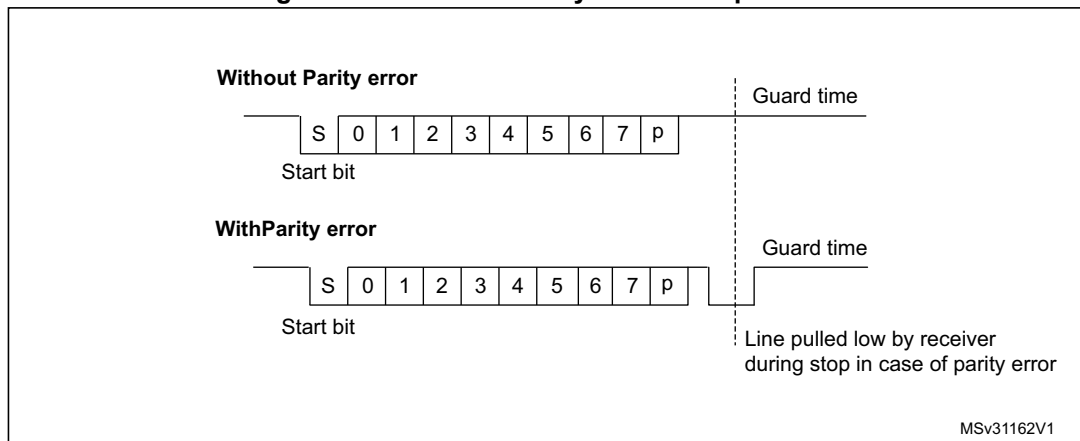
The USART should be configured as:

- 8 bits plus parity: where word length is set to 8 bits and PCE=1 in the USARTx_CR1 register
- 1.5 stop bits: where STOP=11 in the USARTx_CR2 register.

In T=0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

[Figure 329](#) shows examples of what can be seen on the data line with and without parity error.

Figure 329. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol. The number of retries is programmed in the SCARCNT bit field. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit can be set using the TXFRQ bit in the USARTx_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 guard-time). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T=1 mode). If the received character is erroneous, the RXNE/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bit field, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the

desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).

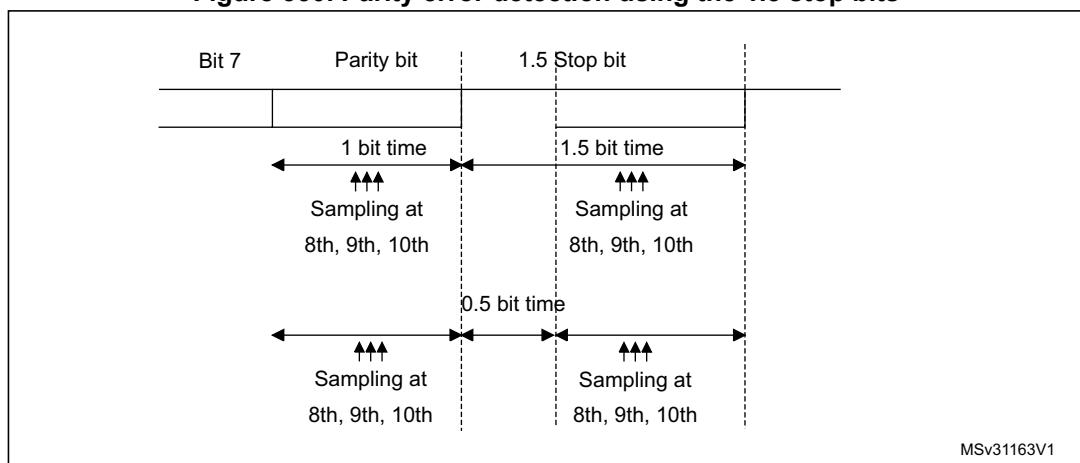
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high.
- The de-assertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: *A break character is not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.*

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 330 details how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 330. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the SCLK output. In Smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the prescaler register USARTx_. SCLK frequency can be programmed from $f_{CK}/2$ to $f_{CK}/62$, where f_{CK} is the peripheral input clock.

Block mode (T=1)

In T=1 (block) mode, the parity error transmission is deactivated, by clearing the NACK bit in the UART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must enable the receiver Timeout feature by setting the RTOEN bit in the USART_CR2 register and program the RTO bits field in 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, the RTOF flag will be set and a timeout interrupt will be generated (if RTOIE bit in the USART_CR1 register is set). If the first character is received before the expiration of the period, it is signaled by the RXNE interrupt.

Note: *The RXNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.*

After the reception of the first character (RXNE interrupt), the RTO bit fields in the RTOR register must be programmed to the CWT (character wait time) - 11 value, in order to allow the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baudtime units. If the smartcard doesn't send a new character in less than the CWT period after the end of the previous character, the USART signals this to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: *The RTO counter starts counting from the end of the first stop bit of the last character in cases STOP = 00, 10. In case of STOP = 11, the RTO counter starts counting 1 bit duration after the beginning of the STOP bit. As in the smartcard protocol definition, the BWT/CWT values are defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT -11 or CWT -11, respectively, taking into account the length of the last character itself.*

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting (TXE=0). The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USARTx_RTOR register. when using DMA mode, before the start of the block, this register field must be programmed to the minimum value (0x0). with this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value will be programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilogue 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 EOBFIE 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: LHH LLL LLH and LHH LHH LLH.

- (H) LHH 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) LHH LHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). when decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHH LLL LLH => the USART received character will be '03' and the parity will be odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card didn't answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it will be correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHH LLL LLH = 0x103 -> inverse convention to be chosen

(H) LHH LHH 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.

29.5.14 IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Please refer to [Section 29.4: USART implementation on page 838](#).

IrDA mode is selected by setting the IREN bit in the USARTx_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USARTx_CR2 register,
- SCEN and HDSEL bits in the USARTx_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 331](#)).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the IrDA decoder), 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 332](#)).
- 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 μ 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 USARTx_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the STOP bits in the USARTx_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 period. 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 USARTx_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 331. IrDA SIR ENDEC- block diagram

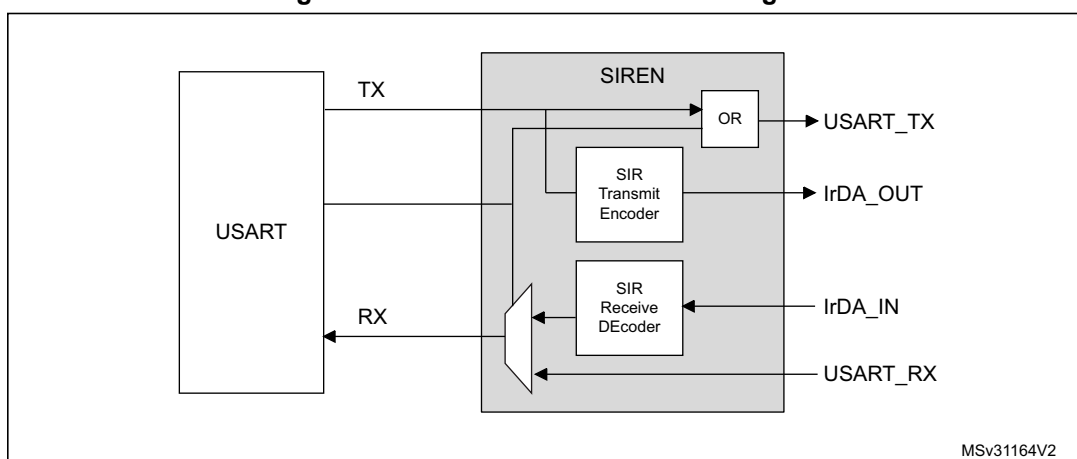
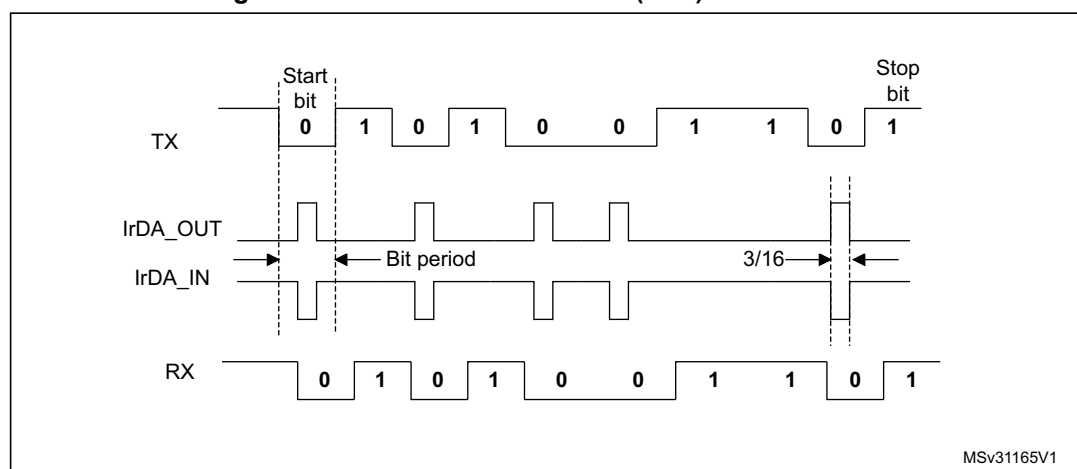


Figure 332. IrDA data modulation (3/16) -Normal Mode



29.5.15 Continuous communication using DMA

The USART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Please refer to [Section 29.4: USART implementation on page 838](#) to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in [Section 29.5.2: Transmitter](#) or [Section 29.5.3: Receiver](#). To perform continuous communication, you can clear the TXE/ RXNE flags in the USARTx_ISR register.

Transmission using DMA

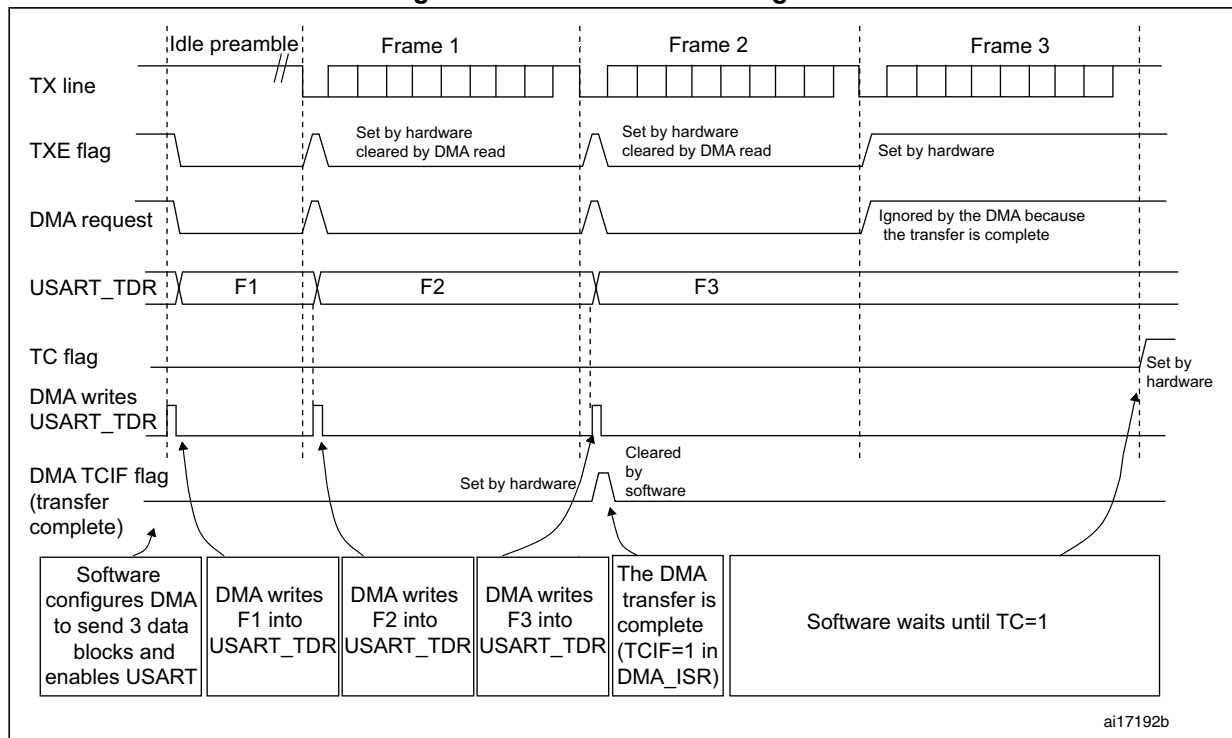
DMA mode can be enabled for transmission by setting DMAT bit in the USARTx_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to [Section 13: Direct memory access controller \(DMA\) on page 244](#)) to the USARTx_TDR register whenever the TXE bit is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USARTx_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USARTx_TDR register from this memory area after each TXE event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USARTx_ISR register by setting the TCCF bit in the USARTx_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or entering Stop mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 333. Transmission using DMA



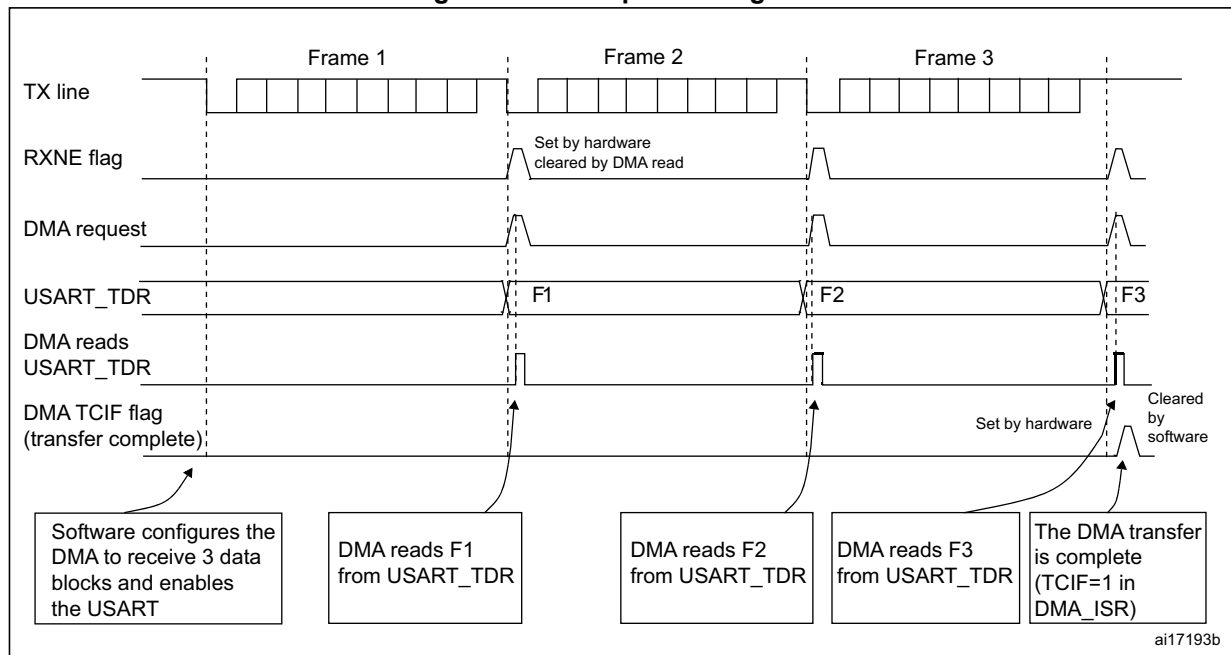
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USARTx_CR3 register. Data is loaded from the USARTx_RDR register to a SRAM area configured using the DMA peripheral (refer to [Section 13: Direct memory access controller \(DMA\) on page 244](#)) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USARTx_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USARTx_RDR to this memory area after each RXNE event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register.
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 334. Reception using DMA



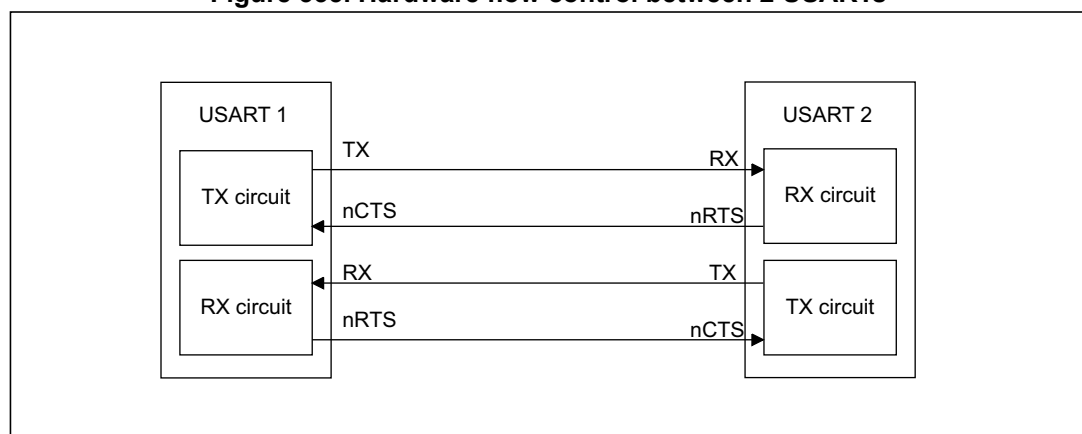
Error flagging and interrupt generation in multibuffer communication

In multibuffer communication if any error occurs during the transaction the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USARTx_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

29.5.16 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The [Figure 335](#) shows how to connect 2 devices in this mode:

Figure 335. Hardware flow control between 2 USARTs

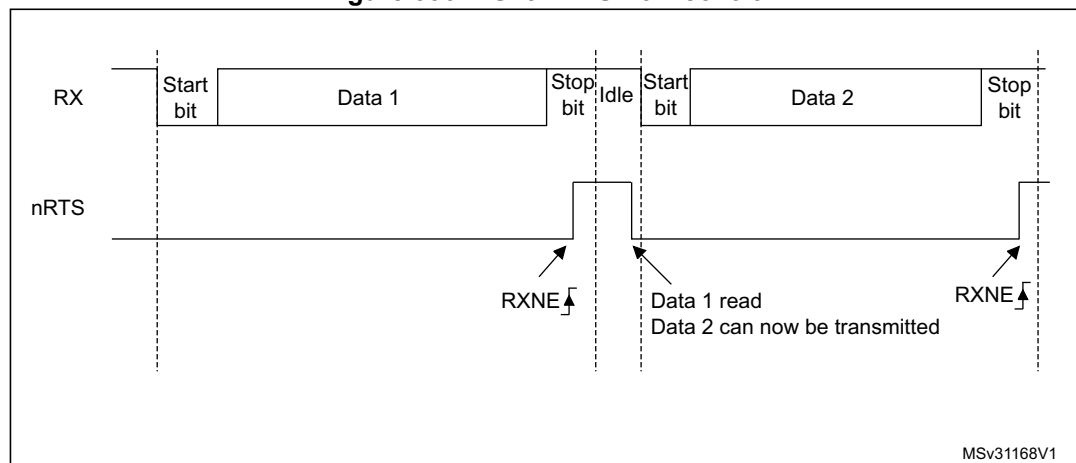


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the USARTx_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, nRTS is de-asserted, indicating that the transmission is expected to stop at the end of the current frame. [Figure 336](#) shows an example of communication with RTS flow control enabled.

Figure 336. RS232 RTS flow control

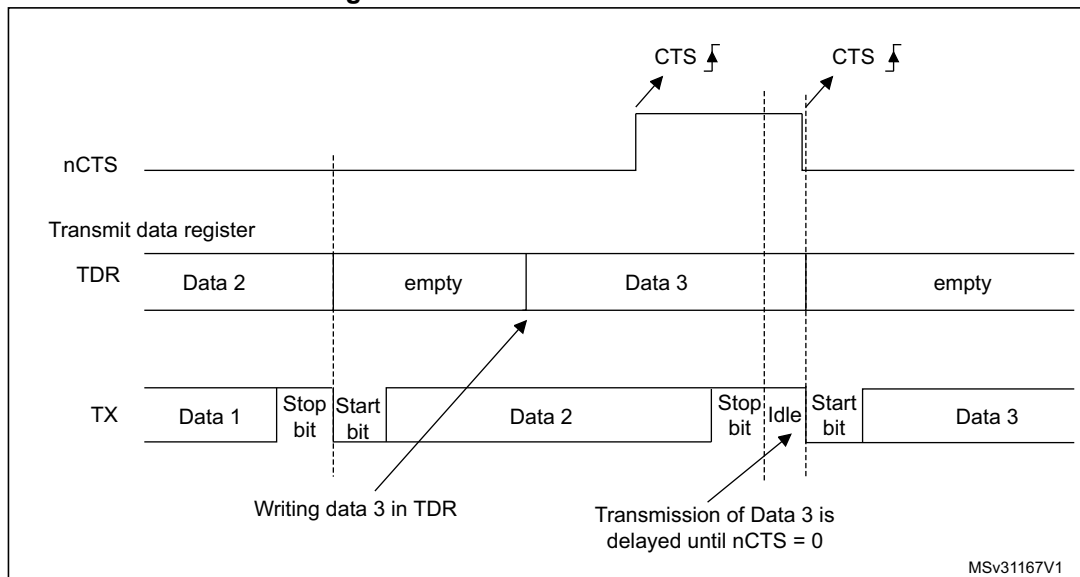


RS232 CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE=0), else the transmission does not occur. When nCTS is de-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 nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USARTx_CR3 register is set. [Figure 337](#) shows an example of communication with CTS flow control enabled.

Figure 337. RS232 CTS flow control



Note: For correct behavior, *nCTS* must be asserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 Driver Enable

The driver enable feature is enabled by setting bit DEM in the USARTx_CR3 control register. This allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the START bit. It is programmed using the DEAT [4:0] bit fields in the USARTx_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bit fields in the USARTx_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USARTx_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit duration, depending on the oversampling rate).

29.5.17 Wakeup from Stop mode

The USART is able to wake up the MCU from Stop mode when the UESM bit is set and the USART clock is set to HSI or LSE (refer to Section Reset and clock control (RCC)).

The MCU wakeup from Stop mode can be done using the standard RXNE interrupt. In this case, the RXNEIE bit must be set before entering Stop mode.

Alternatively, a specific interrupt may be selected through the WUS bit fields.

In order to be able to wake up the MCU from Stop mode, the UESM bit in the USARTx_CR1 control register must be set prior to entering Stop mode.

When the wakeup event is detected, the WUF flag is set by hardware and a wakeup interrupt is generated if the WUFIE bit is set.

Note: Before entering Stop mode, the user must ensure that the USART is not performing a transfer. *BUSY* flag cannot ensure that Stop mode is never entered during a running reception.

The *WUF* flag is set when a wakeup event is detected, independently of whether the MCU is in Stop or in an active mode.

When entering Stop mode just after having initialized and enabled the receiver, the *REACK* bit must be checked to ensure the USART is actually enabled.

When DMA is used for reception, it must be disabled before entering Stop mode and re-enabled upon exit from Stop mode.

The wakeup from Stop mode feature is not available for all modes. For example it doesn't work in SPI mode because the SPI operates in master mode only.

Using Mute mode with Stop mode

If the USART is put into Mute mode before entering Stop mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in Stop mode.
- If the wakeup from Mute mode on address match is used, then the source of wake-up from Stop mode must also be the address match. If the *RXNE* flag is set when entering the Stop mode, the interface will remain in mute mode upon address match and wake up from Stop.
- If the USART is configured to wake up the MCU from Stop mode on *START* bit detection, the *WUF* flag is set, but the *RXNE* flag is not set.

29.6 USART low-power modes

Table 158. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupt causes the device to exit Sleep mode.
Stop	The USART is able to wake up the MCU from Stop mode when the <i>UESM</i> bit is set and the USART clock is set to HSI or LSE. The MCU wakeup from Stop mode can be done using either a standard <i>RXNE</i> or a <i>WUF</i> interrupt.
Standby	The USART is powered down and must be reinitialized when the device has exited from Standby mode.

29.7 USART interrupts

Table 159. USART interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit data register empty	TXE	TXEIE
CTS interrupt	CTSIF	CTSIE
Transmission Complete	TC	TCIE

Table 159. USART interrupt requests (continued)

Interrupt event	Event flag	Enable Control bit
Receive data register not empty (data ready to be read)	RXNE	RXNEIE
Overrun error detected	ORE	
Idle line detected	IDLE	IDLEIE
Parity error	PE	PEIE
LIN break	LBDF	LBDIE
Noise Flag, Overrun error and Framing Error in multibuffer communication.	NF or ORE or FE	EIE
Character match	CMF	CMIE
Receiver timeout error	RTOF	RTOIE
End of Block	EOBF	EOBIE
Wakeup from Stop mode	WUF ⁽¹⁾	WUFIE

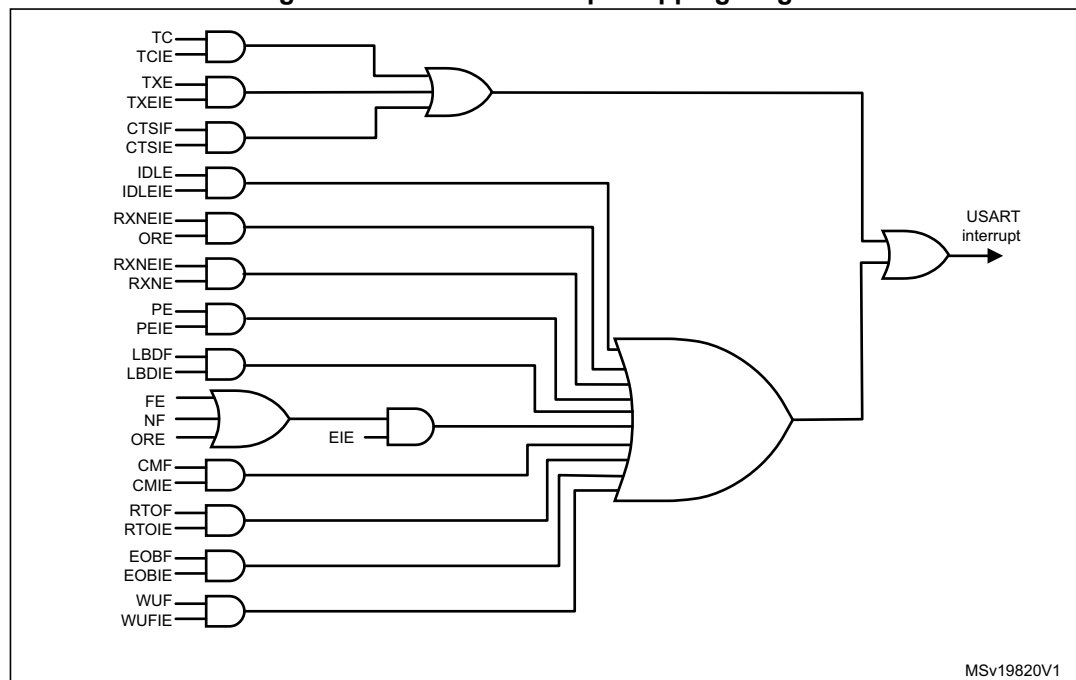
1. The wUF interrupt is active only in Stop mode.

The USART interrupt events are connected to the same interrupt vector (see [Figure 338](#)).

- During transmission: Transmission Complete, Transmit data Register empty or Framing error (in Smartcard mode) interrupt.
- During reception: Idle Line detection, Overrun error, Receive data register not empty, Parity error, LIN break detection, Noise Flag, Framing Error, Character match, etc.

These events generate an interrupt if the corresponding Enable Control Bit is set.

Figure 338. USART interrupt mapping diagram



29.8 USART registers

Refer to [Section 2.1 on page 42](#) for a list of abbreviations used in register descriptions.

29.8.1 Control register 1 (USARTx_CR1)

Address offset: 0x00

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 31:29 Reserved, must be kept at reset value

Bit 28 **M1**: Word length

This bit, with bit 12 (M0), determines the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 data bits, n stop bits

M[1:0] = 10: 1 Start bit, 7 data bits, n stop bits

This bit can only be written when the USART is disabled (UE=0).

Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Autobaudrate (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 is inhibited

1: A USART interrupt is generated when the EOBIF flag is set in the USARTx_ISR register

Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated when the RTOF bit is set in the USARTx_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. [Section 29.4: USART implementation on page 838](#).

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 duration, depending on the oversampling rate).

This bit field 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 cleared. Please refer to [Section 29.4: USART implementation on page 838](#).

Bits 20:16 **DEDT[4:0]**: Driver Enable de-assertion 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 duration, depending on the oversampling rate).

If the USARTx_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bit field 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 cleared. Please refer to [Section 29.4: USART implementation on page 838](#).

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 modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated when the CMF bit is set in the USARTx_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the USART. when set, the USART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit, with bit 28 (M1), determines the word length. It is set or cleared by software. See Bit 28 (M1) description.

This bit can only be written when the USART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bit field can only be written when the USART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bit field can only be written when the USART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.

0: Even parity

1: Odd parity

This bit field can only be written when the USART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever PE=1 in the USARTx_ISR register

Bit 7 **TXEIE**: interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever TXE=1 in the USARTx_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever TC=1 in the USARTx_ISR register

Bit 5 **RXNEIE**: RXNE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever ORE=1 or RXNE=1 in the USARTx_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated whenever IDLE=1 in the USARTx_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 “0” pulse on the TE bit (“0” followed by “1”) sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the USARTx_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**: USART enable in Stop mode

When this bit is cleared, the USART is not able to wake up the MCU from Stop mode.

When this bit is set, the USART is able to wake up the MCU from Stop mode, provided that the USART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from Stop mode.

1: USART able to wake up the MCU from Stop mode. When this function is active, the clock source for the USART must be HSI or LSE (see Section Reset and clock control (RCC)).

Note: It is recommended to set the UESM bit just before entering Stop mode and clear it on exit from Stop mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the USART is kept, but all the status flags, in the USARTx_ISR are set to their default values. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: In order to go into low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the USARTx_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.

29.8.2 Control register 2 (USARTx_CR2)

Address offset: 0x04

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw				

Bits 31:28 **ADD[7:4]**: Address of the USART node

This bit-field gives the address of the USART node or a character code to be recognized.

This is used in multiprocessor communication during Mute mode or Stop mode, for wakeup with 7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. It may also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.

This bit field can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bits 27:24 **ADD[3:0]**: Address of the USART node

This bit-field gives the address of the USART node or a character code to be recognized.

This is used in multiprocessor communication during Mute mode or Stop mode, for wakeup with address mark detection.

This bit field can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USARTx_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 22:21 **ABRM0D[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement. (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bit field 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 forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

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 7/8/9) first, following the start bit.

This bit field can only be written when the USART is disabled (UE=0).

Bit 18 DATAINV: Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)

1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.

This bit field can only be written when the USART is disabled (UE=0).

Bit 17 TXINV: TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)

1: TX pin signal values are inverted. (V_{DD} =0/mark, Gnd=1/idle).

This allows the use of an external inverter on the TX line.

This bit field can only be written when the USART is disabled (UE=0).

Bit 16 RXINV: RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)

1: RX pin signal values are inverted. (V_{DD} =0/mark, Gnd=1/idle).

This allows the use of an external inverter on the RX line.

This bit field can only be written when the USART is disabled (UE=0).

Bit 15 SWAP: Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired connection to another USART.

This bit field can only be written when the USART is disabled (UE=0).

Bit 14 LINEN: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN Sync Breaks (13 low bits) using the SBKRQ bit in the USARTx_RQR register, and to detect LIN Sync breaks.

This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support LIN mode, this bit is reserved and forced by hardware to '0'.

Please refer to [Section 29.4: USART implementation on page 838](#).

Bits 13:12 STOP[1:0]: STOP bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: Reserved

10: 2 stop bits

11: 1.5 stop bits

This bit field can only be written when the USART is disabled (UE=0).

Bit 11 CLKEN: Clock enable

This bit allows the user to enable the SCLK pin.

0: SCLK pin disabled

1: SCLK pin enabled

This bit can only be written when the USART is disabled (UE=0).

Note: If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Note: In order to provide correctly the SCLK clock to the Smartcard when SCLK is always available When CLKEN = 1, regardless of the UE bit value, the steps below must be respected:

- UE = 0
- SCEN = 1
- GTPR configuration (If PSC needs to be configured, it is recommended to configure PSC and GT in a single access to USARTx_ GTPR register).
- CLKEN= 1
- UE = 1

Bit 10 CPOL: Clock polarity

This bit allows the user to select the polarity of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on SCLK pin outside transmission window

1: Steady high value on SCLK pin outside transmission window

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 9 CPHA: Clock phase

This bit is used to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see

[Figure 326](#) and [Figure 327](#))

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 8 LBCL: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the SCLK pin

1: The clock pulse of the last data bit is output to the SCLK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bits in the USARTx_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 forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

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 USARTx_ISR register

Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE=0).

Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to Section 29.4: USART implementation on page 838.

Bit 4 **ADDM7**: 7-bit Address Detection/4-bit Address Detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE=0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

Note: The 3 bits (CPOL, CPHA, LBCL) should not be written while the transmitter is enabled.

29.8.3 Control register 3 (USARTx_CR3)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUFIE		SCARCNT2:0]			Res.
									rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	v	v	rw	rw	rw	rw

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

Bit 23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wakeup from Stop mode interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever WUF=1 in the USARTx_ISR register

Note: WUFIE must be set before entering in Stop mode.

The WUF interrupt is active only in Stop mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.

Bit 21:20 WUS[1:0]: Wakeup from Stop mode interrupt flag selection

This bit-field specify the event which activates the WUF (wakeup from Stop mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WuF active on Start bit detection

11: WUF active on RXNE.

This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.

Bit 19:17 SCARCNT[2:0]: Smartcard auto-retry count

This bit-field specifies the number of retries in transmit and receive, in Smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number of erroneous reception trials, before generating a reception error (RXNE and PE bits set).

This bit field must be programmed only when the USART is disabled (UE=0).

When the USART is enabled (UE=1), this bit field may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmit mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

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 cleared. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 14 DEM: Driver enable mode

This bit allows the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

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 cleared. [Section 29.4: USART implementation on page 838](#).

Bit 13 DDRE: DMA Disable on Reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data will be transferred (used for Smartcard mode).

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the USART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 OVRDIS: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USARTx_RDR register.

This bit can only be written when the USART is disabled (UE=0).

Note: This control bit allows checking the communication flow without reading the data.

Bit 11 ONEBIT: One sample bit method enable

This bit allows the user to select the sample method. When the one sample bit method is selected the noise detection flag (NF) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE=0).

Bit 10 CTSIE: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF=1 in the USARTx_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 9 CTSE: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is de-asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is de-asserted, the transmission is postponed until nCTS is asserted.

This bit can only be written when the USART is disabled (UE=0)

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 8 RTSE: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE=0).

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

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 bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 4 NACK: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bit field can only be written when the USART is disabled (UE=0).

Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 3 HDSEL: Half-duplex selection

Selection of Single-wire Half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the USART is disabled (UE=0).

Bit 2 IRLP: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 1 IREN: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 0 EIE: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NF=1 in the USARTx_ISR register).

0: Interrupt is inhibited

1: An interrupt is generated when FE=1 or ORE=1 or NF=1 in the USARTx_ISR register.

29.8.4 Baud rate register (USARTx_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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 15:4 **BRR[15:4]**

BRR[15:4] = USARTDIV[15:4]

Bits 3:0 **BRR[3:0]**

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

29.8.5 Guard time and prescaler register (USARTx_GTPR)

Address offset: 0x10

Reset value: 0x0000

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

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

Bits 15:8 **GT[7:0]**: Guard time value

This bit-field is used to program the Guard time value in terms of number of baud clock periods.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

This bit field can only be written when the USART is disabled (UE=0).

Note: If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bits 7:0 **PSC[7:0]**: Prescaler value

In IrDA Low-power and normal IrDA mode:

PSC[7:0] = IrDA Normal and Low-Power Baud Rate

Used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

In Smartcard mode:

PSC[4:0]: Prescaler value

Used for programming the prescaler for dividing the USART source clock to provide the Smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bit field can only be written when the USART is disabled (UE=0).

Note: Bits [7:5] must be kept cleared if Smartcard mode is used.

This bit field is reserved and forced by hardware to '0' when the Smartcard and IrDA modes are not supported. Please refer to [Section 29.4: USART implementation on page 838](#).

29.8.6 Receiver timeout register (USARTx_RTOR)

Address offset: 0x14

Reset value: 0x0000

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

Bits 31:24 **BLLEN[7:0]**: Block Length

This bit-field gives the Block length in Smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

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.

This bit-field can be used also in other modes. In this case, the Block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bit-field gives the Receiver timeout value in terms of number of bit duration.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard chapter for more details.

In this case, the timeout measurement is done starting from the Start Bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on the fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the Receiver timeout feature is not supported. Please refer to [Section 29.4: USART implementation on page 838](#).

29.8.7 Request register (USARTx_RQR)

Address offset: 0x18

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value

Bit 4 **TXFRQ**: Transmit data flush request

Writing 1 to this bit sets the TXE flag.

This allows to discard the transmit data. This bit must be used only in Smartcard mode, when data has not been sent due to errors (NACK) and the FE flag is active in the USARTx_ISR register.

If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This allows to discard the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and sets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: In the case 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 flag in the USARTx_ISR and request an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

29.8.8 Interrupt & status register (USARTx_ISR)

Address offset: 0x1C

Reset value: 0x00C0

31	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
ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r

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

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 Stop mode.

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.

Bit 21 TEACK: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USARTx_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 WUF: Wakeup from Stop mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bit field. It is cleared by software, writing a 1 to the WUCF in the USARTx_ICR register. An interrupt is generated if WUFIE=1 in the USARTx_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

The WUF interrupt is active only in Stop mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.

Bit 19 RWU: Receiver wakeup from Mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USARTx_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USARTx_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Bit 18 SBKF: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USARTx_RQR register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character is transmitted

1: Break character will be transmitted

Bit 17 CMF: Character match flag

This bit is set by hardware, when the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USARTx_ICR register.

An interrupt is generated if CMIE=1 in the USARTx_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 BUSY: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 ABRF: Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE will also be set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case). It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USARTx_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'.

Bit 14 ABRE: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed).

It is cleared by software, by writing 1 to the ABRRQ bit in the USARTx_CR3 register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'.

Bit 13 Reserved, must be kept at reset value.**Bit 12 EOBF:** End of block flag

This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBI=1 in the USARTx_CR2 register.

It is cleared by software, writing 1 to the EOBCF in the USARTx_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

Note: If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

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 USARTx_ICR register.

An interrupt is generated if RTOIE=1 in the USARTx_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 will be set.

If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'.

Bit 10 CTS: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set

1: nCTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 9 CTSIF: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USARTx_ICR register.

An interrupt is generated if CTSIE=1 in the USARTx_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 8 LBDF: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCE in the USARTx_ICR.

An interrupt is generated if LBDIE = 1 in the USARTx_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 7 TXE: Transmit data register empty

This bit is set by hardware when the content of the USARTx_TDR register has been transferred into the shift register. It is cleared by a write to the USARTx_TDR register.

The TXE flag can also be cleared by writing 1 to the TXFRQ in the USARTx_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 USARTx_CR1 register.

0: data is not transferred to the shift register

1: data is transferred to the shift register

Note: This bit is used during single buffer transmission.

Bit 6 TC: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE=1 in the USARTx_CR1 register. It is cleared by software, writing 1 to the TCCF in the USARTx_ICR register or by a write to the USARTx_TDR register.

An interrupt is generated if TCIE=1 in the USARTx_CR1 register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit will be set immediately.

Bit 5 RXNE: Read data register not empty

This bit is set by hardware when the content of the RDR shift register has been transferred to the USARTx_RDR register. It is cleared by a read to the USARTx_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USARTx_RQR register.

An interrupt is generated if RXNEIE=1 in the USARTx_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 USARTx_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USARTx_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit will not be set again until the RXNE bit has been set (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 RDR register while RXNE=1. It is cleared by a software, writing 1 to the ORECF, in the USARTx_ICR register.

An interrupt is generated if RXNEIE=1 or EIE = 1 in the USARTx_CR1 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multibuffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the OVRDIS bit is set in the USARTx_CR3 register.

Bit 2 NF: START bit 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 NFCF bit in the USARTx_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 NF flag is set during multibuffer communication if the EIE bit is set.

Note: When the line is noise-free, the NF flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to [Section 29.5.5: Tolerance of the USART receiver to clock deviation on page 854](#)).

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 USARTx_ICR register. In Smartcard mode, in transmission, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USARTx_CR1 register.

0: No Framing error is detected

1: Framing error or break character is detected

Bit 0 PE: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USARTx_ICR register.

An interrupt is generated if PEIE = 1 in the USARTx_CR1 register.

0: No parity error

1: Parity error

29.8.9 Interrupt flag clear register (USARTx_ICR)

Address offset: 0x20

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	EOBCF	RTOCF	Res.	CTSCF	LBDCF	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
			w	w		w	w		w		w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from Stop mode clear flag

Writing 1 to this bit clears the WUF flag in the USARTx_ISR register.

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.

Bit 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USARTx_ISR register.

Bit 16:13 Reserved, must be kept at reset value.

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USARTx_ISR register.

Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 11 **RTOCF**: Receiver timeout clear flag

Writing 1 to this bit clears the RTOF flag in the USARTx_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

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 USARTx_ISR register.

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 8 **LBDCF**: LIN break detection clear flag

Writing 1 to this bit clears the LBDF flag in the USARTx_ISR register.

Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 29.4: USART implementation on page 838](#).

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the USARTx_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 USARTx_ISR register.

Bit 3 **ORECF**: Overrun error clear flag

Writing 1 to this bit clears the ORE flag in the USARTx_ISR register.

Bit 2 **NCF**: Noise detected clear flag

Writing 1 to this bit clears the NF flag in the USARTx_ISR register.

Bit 1 **FECF**: Framing error clear flag

Writing 1 to this bit clears the FE flag in the USARTx_ISR register.

Bit 0 **PECF**: Parity error clear flag

Writing 1 to this bit clears the PE flag in the USARTx_ISR register.

29.8.10 Receive data register (USARTx_RDR)

Address offset: 0x24

Reset value: Undefined

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

29.8.11 Transmit data register (USARTx_TDR)

Address offset: 0x28

Reset value: Undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 314](#)).

When transmitting with the parity enabled (PCE bit set to 1 in the USARTx_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE=1.

29.8.12 USART register map

The table below gives the USART register map and reset values.

Table 160. USART register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x00	USARTx_CR1	Res.	Res.	Res.	M1	EOBIE	RTOIE	DEAT4	DEAT3	DEAT2	DEAT1	DEAT0	DEDT4	DEDT3	DEDT2	DEDT1	DEDT0	OVER8	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	0		
0x04	USARTx_CR2	ADD[7:4]			ADD[3:0]				RTOEN	ABRMOD1	ABRMOD0	ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	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							
0x08	USARTx_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUFIE	SCARCNT[2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	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		
0x0C	USARTx_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	0	0		
0x10	USARTx_GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]					PSC[7:0]												
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x14	USARTx_RTOR	BLEN[7:0]								RTO[23:0]																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x18	USARTx_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.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ			
	Reset value																											0	0	0	0	0	0		
0x1C	USARTx_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE		
	Reset value										0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	1	0	0	0	0	0	0		

Table 160. USART 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	USARTx_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	EOBCF	RTOCF	Res.	CTSCF	LBDCF	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
	Reset value												0			0					0	0		0	0		0		0	0	0	0	0
0x24	USARTx_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								X	X	X	X	X	X	X	X	X
0x28	USARTx_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								X	X	X	X	X	X	X	X	X

Refer to [Section 3.2](#) for the register boundary addresses.

30 Serial peripheral interface / inter-IC sound (SPI/I2S)

30.1 Introduction

The SPI/I²S interface can be used to communicate with external devices using the SPI protocol or the I²S audio protocol. SPI or I²S mode is selectable by software. SPI Motorola mode is selected by default after a device reset.

The serial peripheral interface (SPI) protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

The Inter-IC sound (I²S) protocol is also a synchronous serial communication interface. It can operate in slave or master mode with full duplex and half-duplex communication. It can address four different audio standards including the Philips I²S standard, the MSB- and LSB-justified standards and the PCM standard.

30.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4-bit to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to $f_{PCLK}/2$.
- Slave mode frequency up to $f_{PCLK}/2$.
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
 - CRC value can be transmitted as last byte in Tx mode
 - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- SPI TI mode support

30.3 I2S main features

- Full duplex communication
- 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 I²S protocols:
 - I²S Philips standard
 - MSB-Justified standard (Left-Justified)
 - LSB-Justified standard (Right-Justified)
 - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for transmission and reception (16-bit wide)
- Master clock can be output to drive an external audio component. Ratio is fixed at $256 \times F_S$ (where F_S is the audio sampling frequency)
- I²S (I2S2 and I2S3) clock can be derived from an external clock mapped on the I2S_CKIN pin.

30.4 SPI/I2S implementation

This manual describes the full set of features implemented in SPI1, SPI2 and SPI3.

Table 161. STM32F302x6/8 SPI implementation

SPI Features ⁽¹⁾	SPI2	SPI3
Hardware CRC calculation	X	X
Rx/Tx FIFO	X	X
NSS pulse mode	X	X
I2S mode	X	X
TI mode	X	X

1. X = supported.

Table 162. STM32F302xB/C/D/E SPI implementation

SPI Features ⁽¹⁾	SPI1	SPI2	SPI3	SPI4 ⁽²⁾
Hardware CRC calculation	X	X	X	X
Rx/Tx FIFO	X	X	X	X
NSS pulse mode	X	X	X	X
I2S mode	-	X	X	-
TI mode	X	X	X	X

1. X = supported.

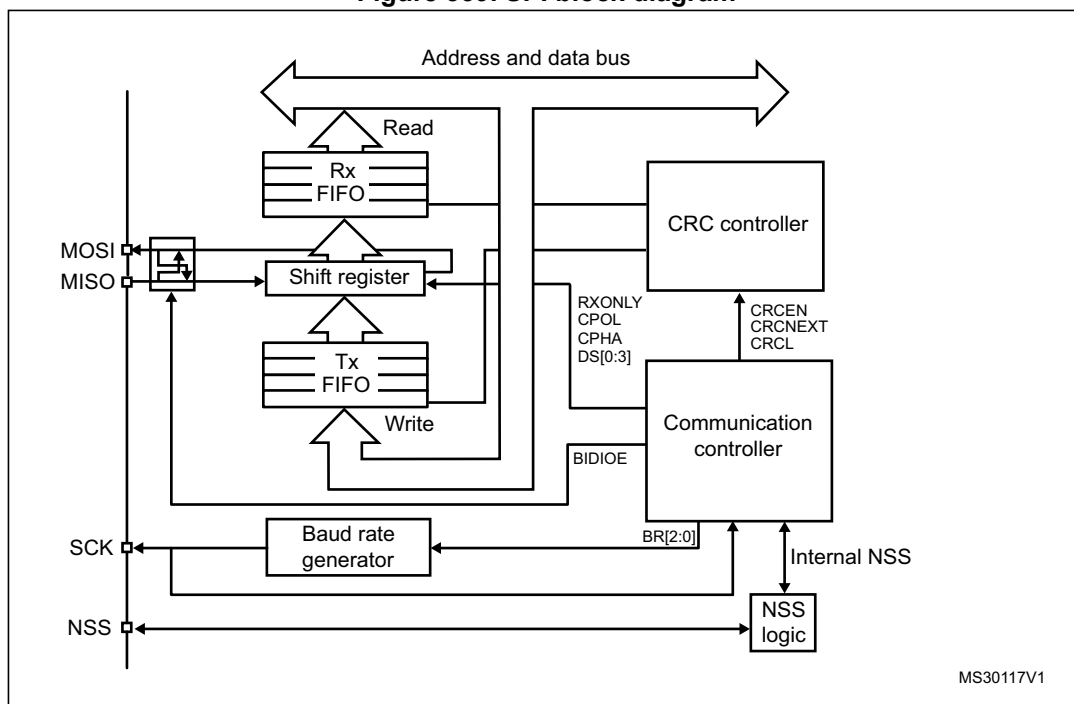
2. SPI4 is only in STM32F302xD/E.

30.5 SPI functional description

30.5.1 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 339](#).

Figure 339. 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 30.5.4: Slave select \(NSS\) pin management](#) for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

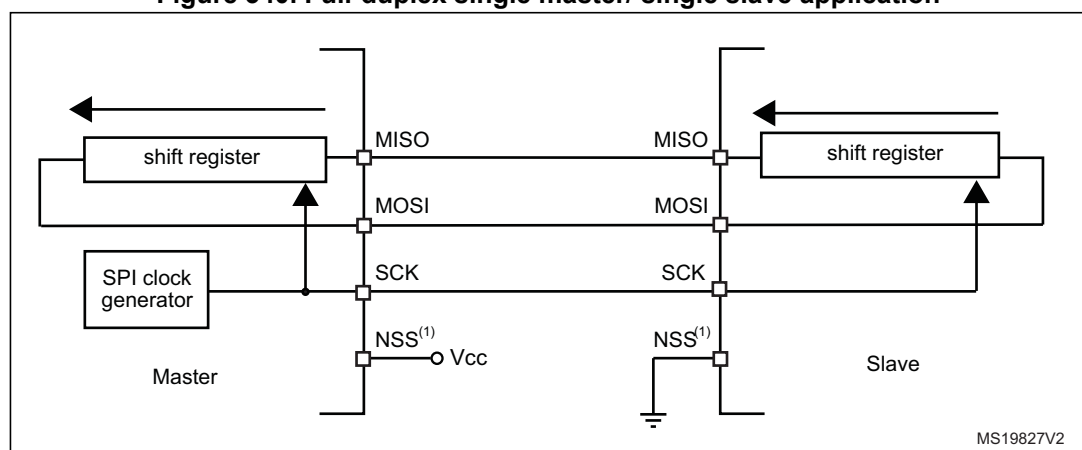
30.5.2 Communications between one master and one slave

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 340. Full-duplex single master/ single slave application

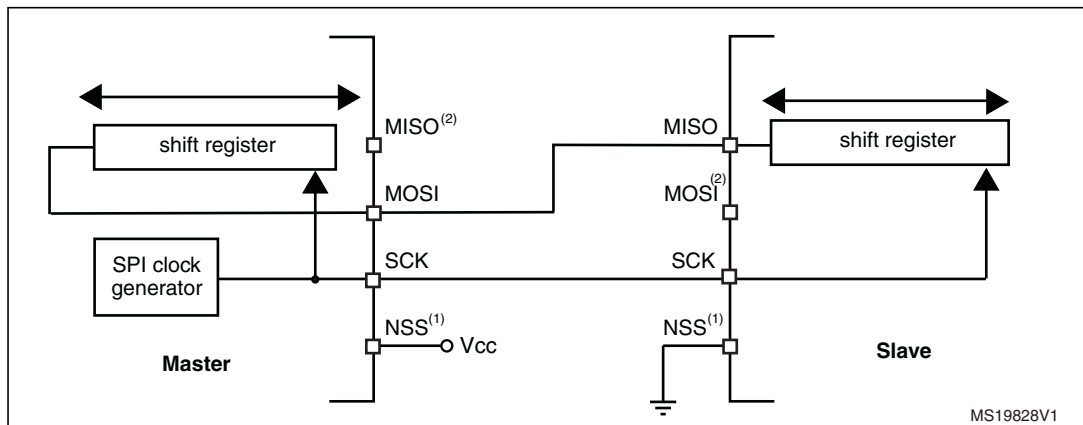


1. The NSS pin is configured as an input in this case.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting the BIDIMODE bit in the SPIx_CR1 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the BDIOE bit in their SPIx_CR1 registers. In this configuration, the master's MISO pin and the slave's MOSI pin are free for other application uses and act as GPIOs.

Figure 341. Half-duplex single master/ single slave application

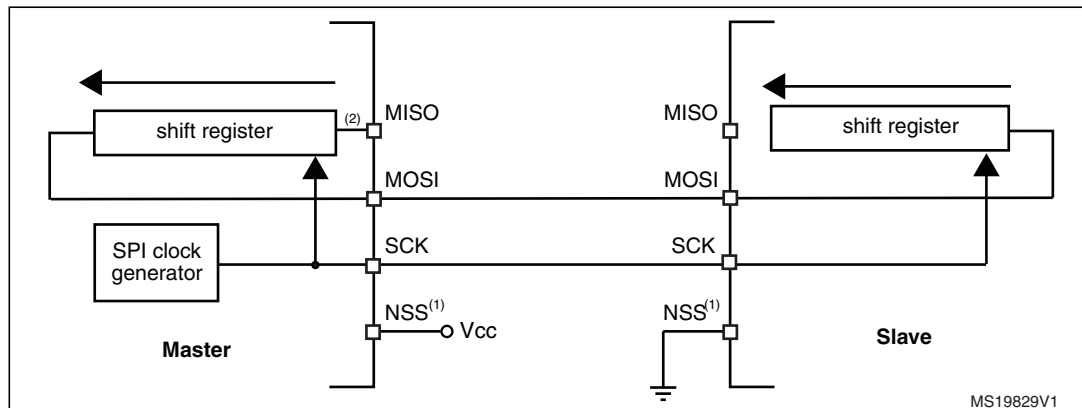


1. The NSS pin is configured as an input in this case.
2. In this configuration, the master's MISO pin and the slave's MOSI pin can be used as GPIOs.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx_CR2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0):** The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.
- **Receive-only mode (RXONLY=1):** The application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see [30.5.4: Slave select \(NSS\) pin management](#)). Received data events appear depending on the data buffer configuration. In the master configuration, the MOSI output is disabled and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled. The only way to stop the clock is to clear the RXONLY bit or the SPE bit and wait until the incoming pattern from the MISO pin is finished and fills the data buffer structure, depending on its configuration.

Figure 342. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)

1. The NSS pin is configured as an input in this case.
2. The input information is captured in the shift register and must be ignored in standard transmit only mode (for example, OVF flag)
3. In this configuration, both the MISO pins can be used as GPIOs.

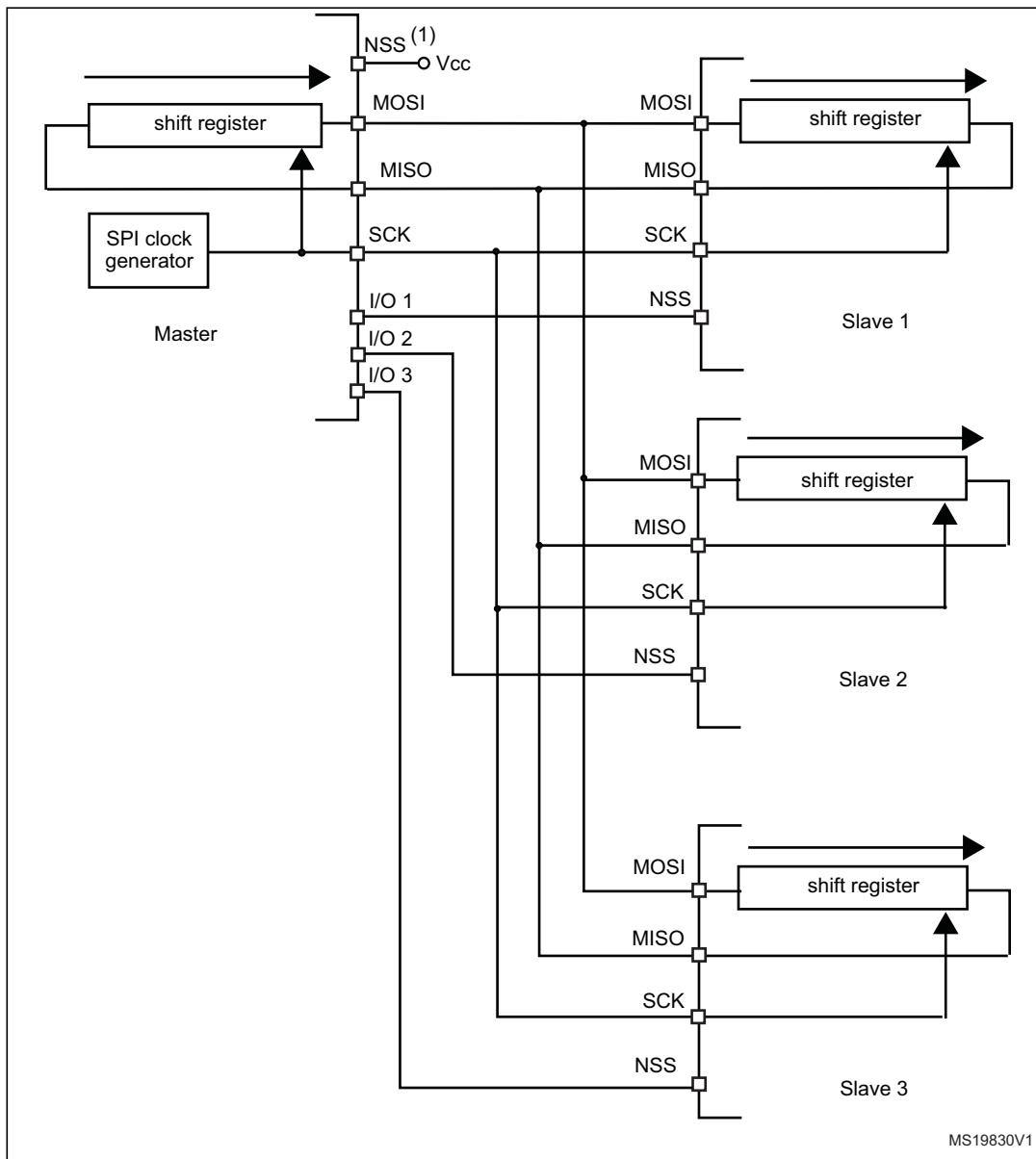
Note:

Any simplex communication can be alternatively replaced by a variant of the half duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).

30.5.3 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses GPIO pins to manage the chip select lines for each slave (see [Figure 343](#)). 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 343. Master and three independent slaves



1. As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see [Section 11.3.7: I/O alternate function input/output on page 221](#)).

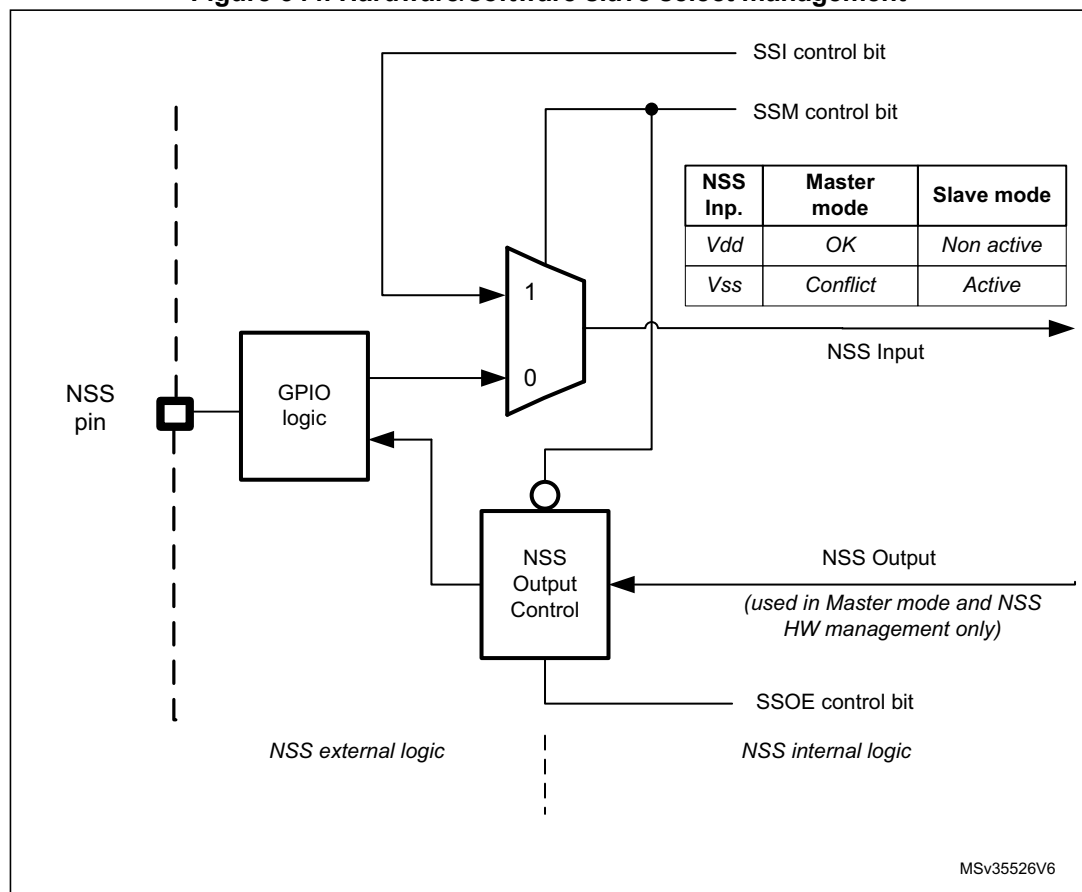
30.5.4 Slave select (NSS) pin management

In slave mode, the NSS works as a standard “chip select” input and lets the slave communicate with the master. In master mode, NSS can be used either as output or input. As an input it can prevent multimaster bus collision, and as an output it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SSM bit in the SPIx_CR1 register:

- **Software NSS management (SSM = 1):** in this configuration, slave select information is driven internally by the SSI bit value in register SPIx_CR1. The external NSS pin is free for other application uses.
- **Hardware NSS management (SSM = 0):** in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in register SPIx_CR1).
 - **NSS output enable (SSM=0,SSOE = 1):** this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (SPE=1), and is kept low until the SPI is disabled (SPE =0). A pulse can be generated between continuous communications if NSS pulse mode is activated (NSSP=1). The SPI cannot work in multimaster configuration with this NSS setting.
 - **NSS output disable (SSM=0, SSOE = 0):** if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters master mode fault state and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while NSS line is at low level.

Figure 344. Hardware/software slave select management



30.5.5 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx_CR1 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

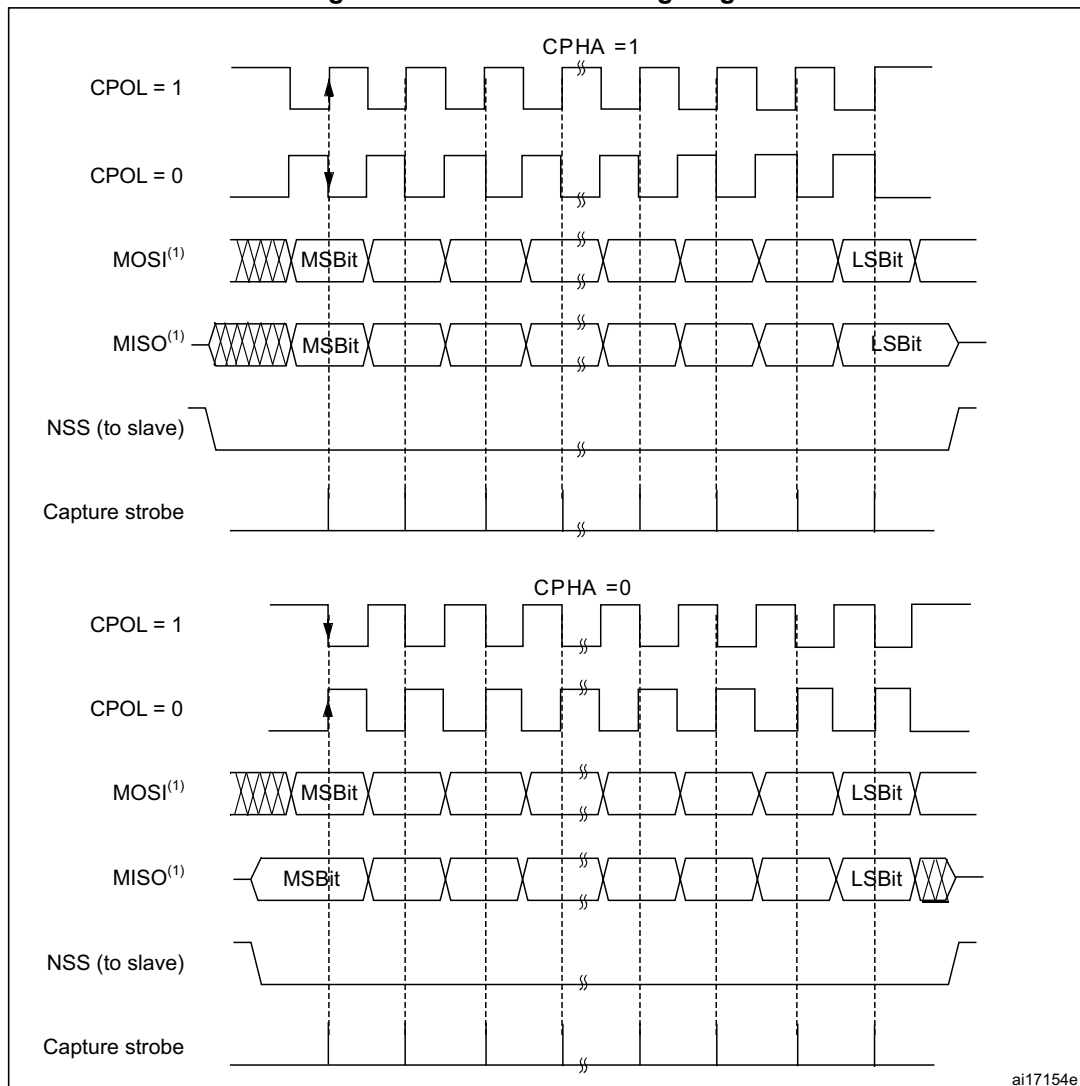
If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 345, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

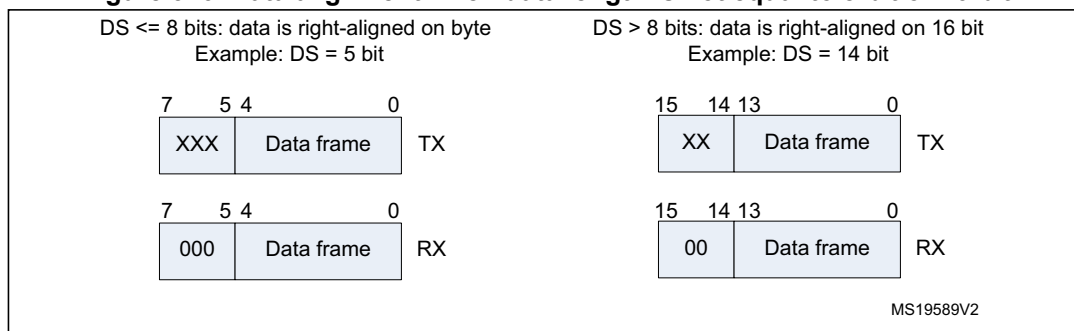
Note: *Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit. The idle state of SCK must correspond to the polarity selected in the SPIx_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).*

Figure 345. Data clock timing diagram



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 346](#)). During communication, only bits within the data frame are clocked and transferred.

Figure 346. 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.

30.5.6 Configuration of SPI

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated chapters. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI_CR1 register:
 - a) Configure the serial clock baud rate using the BR[2:0] bits (**Note:** 4).
 - b) Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock (CPHA must be cleared in NSSP mode). (**Note:** 2).
 - c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE can't be set at the same time).
 - d) Configure the LSBFIRST bit to define the frame format (**Note:** 2).
 - e) Configure the CRCL and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
 - f) Configure SSM and SSI (**Note:** 2 & 3).
 - g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
3. Write to SPI_CR2 register:
 - a) Configure the DS[3:0] bits to select the data length for the transfer.
 - b) Configure SSOE (**Note:** 1 & 2 & 3).
 - c) Set the FRF bit if the TI protocol is required (keep NSSP bit cleared in TI mode).
 - d) Set the NSSP bit if the NSS pulse mode between two data units is required (keep 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

30.5.7 Procedure for enabling SPI

It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

The master at full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled and TXFIFO is not empty, or with the next write to TXFIFO.

In any master receive only mode (RXONLY=1 or BIDIMODE=1 & BIDIOE=0), master starts to communicate and the clock starts running immediately after SPI is enabled.

For handling DMA, follow the dedicated chapter.

30.5.8 Data transmission and reception procedures

RXFIFO and TXFIFO

All SPI data transactions pass through the 32-bit embedded FIFOs. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each direction has its own FIFO called TXFIFO and RXFIFO. These FIFOs are used in all SPI modes except for receiver-only mode (slave or master) with CRC calculation enabled (see [Section 30.5.13: CRC calculation](#)).

The handling of FIFOs depends on the data exchange mode (duplex, simplex), data frame format (number of bits in the frame), access size performed on the FIFO data registers (8-bit or 16-bit), and whether or not data packing is used when accessing the FIFOs (see [Section 30.5.12: TI mode](#)).

A read access to the SPIx_DR register returns the oldest value stored in RXFIFO that has not been read yet. A write access to the SPIx_DR stores the written data in the TXFIFO at the end of a send queue. The read access must be always aligned with the RXFIFO threshold configured by the FRXTH bit in SPIx_CR2 register. FTLVL[1:0] and FRLVL[1:0] bits indicate the current occupancy level of both FIFOs.

A read access to the SPIx_DR register must be managed by the RXNE event. This event is triggered when data is stored in RXFIFO and the threshold (defined by FRXTH bit) is reached. When RXNE is cleared, RXFIFO is considered to be empty. In a similar way, write access of a data frame to be transmitted is managed by the TXE event. This event is triggered when the TXFIFO level is less than or equal to half of its capacity. Otherwise TXE is cleared and the TXFIFO is considered as full. In this way, RXFIFO can store up to four data frames, whereas TXFIFO can only store up to three when the data frame format is not greater than 8 bits. This difference prevents possible corruption of 3x 8-bit data frames already stored in the TXFIFO when software tries to write more data in 16-bit mode into

TXFIFO. Both TXE and RXNE events can be polled or handled by interrupts. See [Figure 348](#) through [Figure 351](#).

Another way to manage the data exchange is to use DMA (see [Section 13.2: DMA main features](#)).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at [Section 30.5.9: SPI status flags](#)). An overrun event can be polled or handled by an interrupt.

The BSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

Sequence handling

A few data frames can be passed at single sequence to complete a message. When transmission is enabled, a sequence begins and continues while any data is present in the TXFIFO of the master. The clock signal is provided continuously by the master until TXFIFO becomes empty, then it stops waiting for additional data.

In receive-only modes, half duplex (BIDIMODE=1, BIDIOE=0) or simplex (BIDIMODE=0, RXONLY=1) the master starts the sequence immediately when both SPI is enabled and receive-only mode is activated. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no underflow error signal for master or slave in SPI mode, and data from the slave is always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislave system to select just one of the slaves for communication. In a single slave system it is not necessary to control the slave with NSS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. NSS can be managed by both software and hardware (see [Section 30.5.4: Slave select \(NSS\) pin management](#)).

When the BSY bit is set it signifies an ongoing data frame transaction. When the dedicated frame transaction is finished, the RXNE flag is raised. The last bit is just sampled and the complete data frame is stored in the RXFIFO.

Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Ongoing transactions can be corrupted in this case. In some modes the disable procedure is the only way to stop continuous communication running.

Master in full duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. Special care must be taken in packing mode when an odd number of data frames are transacted to

prevent some dummy byte exchange (refer to [Data packing](#) section). Before the SPI is disabled in these modes, the user must follow standard disable procedure. When the SPI is disabled at the master transmitter while a frame transaction is ongoing or next data frame is stored in TXFIFO, the SPI behavior is not guaranteed.

When the master is in any receive only mode, the only way to stop the continuous clock is to disable the peripheral by SPE=0. This must occur in specific time window within last data frame transaction just between the sampling time of its first bit and before its last bit transfer starts (in order to receive a complete number of expected data frames and to prevent any additional “dummy” data reading after the last valid data frame). Specific procedure must be followed when disabling SPI in this mode.

Data received but not read remains stored in RXFIFO when the SPI is disabled, and must be processed the next time the SPI is enabled, before starting a new sequence. To prevent having unread data, ensure that RXFIFO is empty when disabling the SPI, by using the correct disabling procedure, or by initializing all the SPI registers with a software reset via the control of a specific register dedicated to peripheral reset (see the SPIiRST bits in the RCC_APB1RSTR registers).

Standard disable procedure is based on pulling BSY status together with FTLVL[1:0] to check if a transmission session is fully completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When NSS signal is managed by software and master has to provide proper end of NSS pulse for slave, or
- When transactions' streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure is (except when receive only mode is used):

1. Wait until FTLVL[1:0] = 00 (no more data to transmit).
2. Wait until BSY=0 (the last data frame is processed).
3. Disable the SPI (SPE=0).
4. Read data until FRLVL[1:0] = 00 (read all the received data).

The correct disable procedure for certain receive only modes is:

1. Interrupt the receive flow by disabling SPI (SPE=0) in the specific time window while the last data frame is ongoing.
2. Wait until BSY=0 (the last data frame is processed).
3. Read data until FRLVL[1:0] = 00 (read all the received data).

Note: *If packing mode is used and an odd number of data frames with a format less than or equal to 8 bits (fitting into one byte) has to be received, FRXTH must be set when FRLVL[1:0] = 01, in order to generate the RXNE event to read the last odd data frame and to keep good FIFO pointer alignment.*

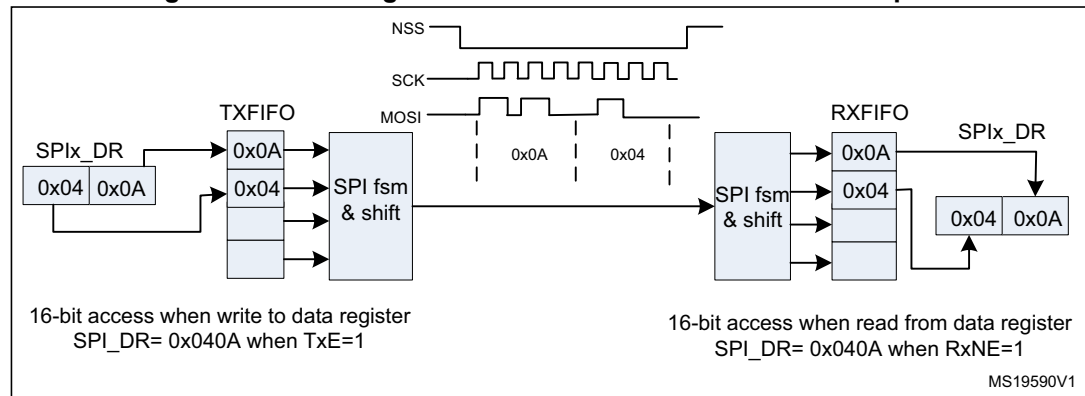
Data packing

When the data frame size fits into one byte (less than or equal to 8 bits), data packing is used automatically when any read or write 16-bit access is performed on the SPIx_DR register. The double data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other half stored in the MSB. [Figure 347](#) 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 347. Packing data in FIFO for transmission and reception



Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXE or RXNE enable bit in the SPIx_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx_DR register.

See [Figure 348](#) through [Figure 351](#).

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 915](#).)

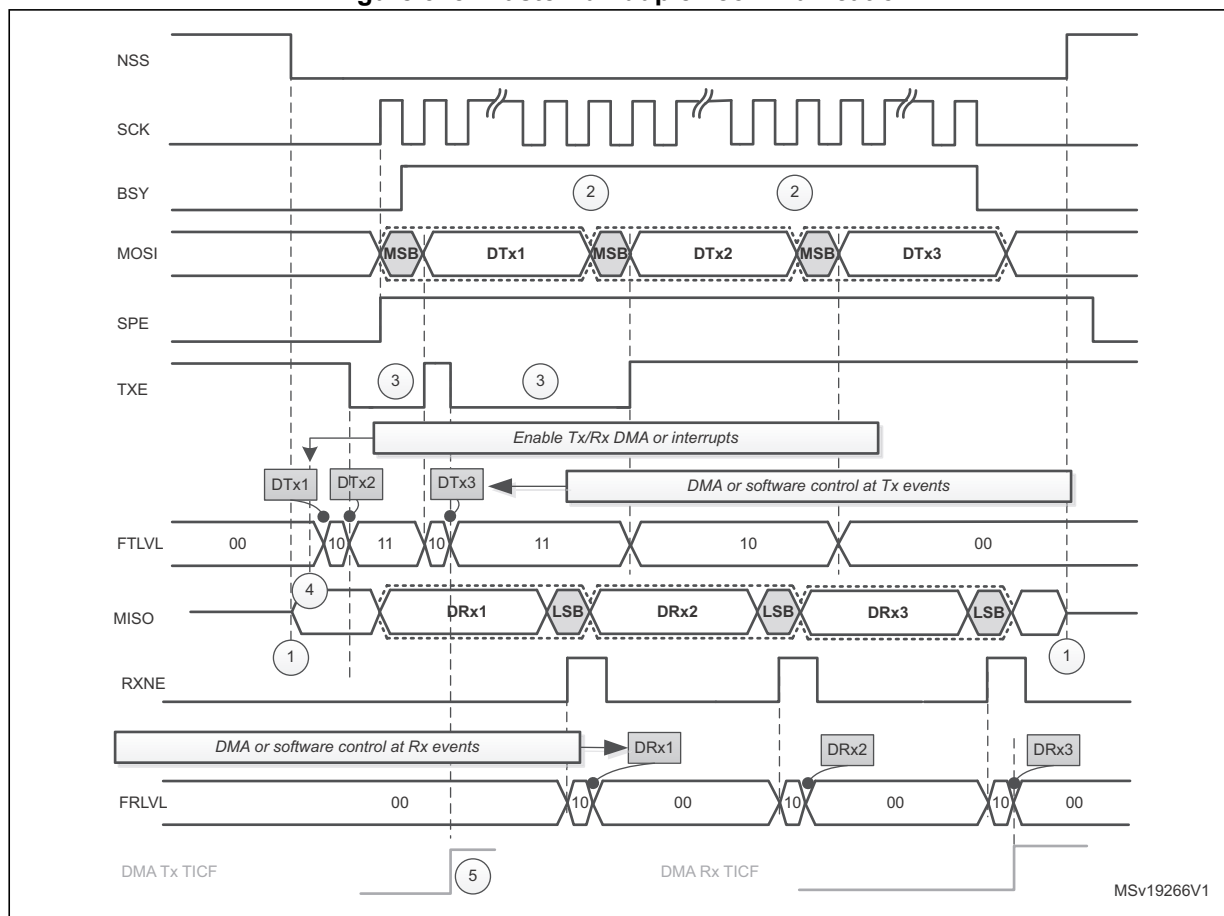
Communication diagrams

Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by pulling, interrupts or DMA. For simplicity, the LSBFIRST=0, CPOL=0 and CPHA=1 setting is used as a common assumption here. No complete configuration of DMA streams is provided.

The following numbered notes are common for [Figure 348 on page 919](#) through [Figure 351 on page 922](#).

1. The slave starts to control MISO line as NSS is active and SPI is enabled, and is disconnected from the line when one of them is released. Sufficient time must be provided for the slave to prepare data dedicated to the master in advance before its transaction starts.
At the master, the SPI peripheral takes control at MOSI and SCK signals (occasionally at NSS signal as well) only if SPI is enabled. If SPI is disabled the SPI peripheral is disconnected from GPIO logic, so the levels at these lines depends on GPIO setting exclusively.
2. At the master, BSY stays active between frames if the communication (clock signal) is continuous. At the slave, BSY signal always goes down for at least one clock cycle between data frames.
3. The TXE signal is cleared only if TXFIFO is full.
4. The DMA arbitration process starts just after the TXDMAEN bit is set. The TXE interrupt is generated just after the TXEIE is set. As the TXE signal is at an active level, data transfers to TxFIFO start, until TxFIFO becomes full or the DMA transfer completes.
5. If all the data to be sent can fit into TxFIFO, the DMA Tx TCIF flag can be raised even before communication on the SPI bus starts. This flag always rises before the SPI transaction is completed.
6. The CRC value for a package is calculated continuously frame by frame in the SPIx_TxCRCR and SPIx_RxCRCR registers. The CRC information is processed after the entire data package has completed, either automatically by DMA (Tx channel must be set to the number of data frames to be processed) or by SW (the user must handle CRCNEXT bit during the last data frame processing).
While the CRC value calculated in SPIx_TxCRCR is simply sent out by transmitter, received CRC information is loaded into Rx FIFO and then compared with the SPIx_RxCRCR register content (CRC error flag can be raised here if any difference). This is why the user must take care to flush this information from the FIFO, either by software reading out all the stored content of Rx FIFO, or by DMA when the proper number of data frames is preset for Rx channel (number of data frames + number of CRC frames) (see the settings at the example assumption).
7. In data packed mode, TxE and RxNE events are paired and each read/write access to the FIFO is 16 bits wide until the number of data frames are even. If the Tx FIFO is $\frac{3}{4}$ full FTLVL status stays at FIFO full level. That is why the last odd data frame cannot be stored before the Tx FIFO becomes $\frac{1}{2}$ full. This frame is stored into Tx FIFO with an 8-bit access either by software or automatically by DMA when LDMA_TX control is set.
8. To receive the last odd data frame in packed mode, the Rx threshold must be changed to 8-bit when the last data frame is processed, either by software setting FRXTH=1 or automatically by a DMA internal signal when LDMA_RX is set.

Figure 348. Master full duplex communication



Assumptions for master full duplex communication example:

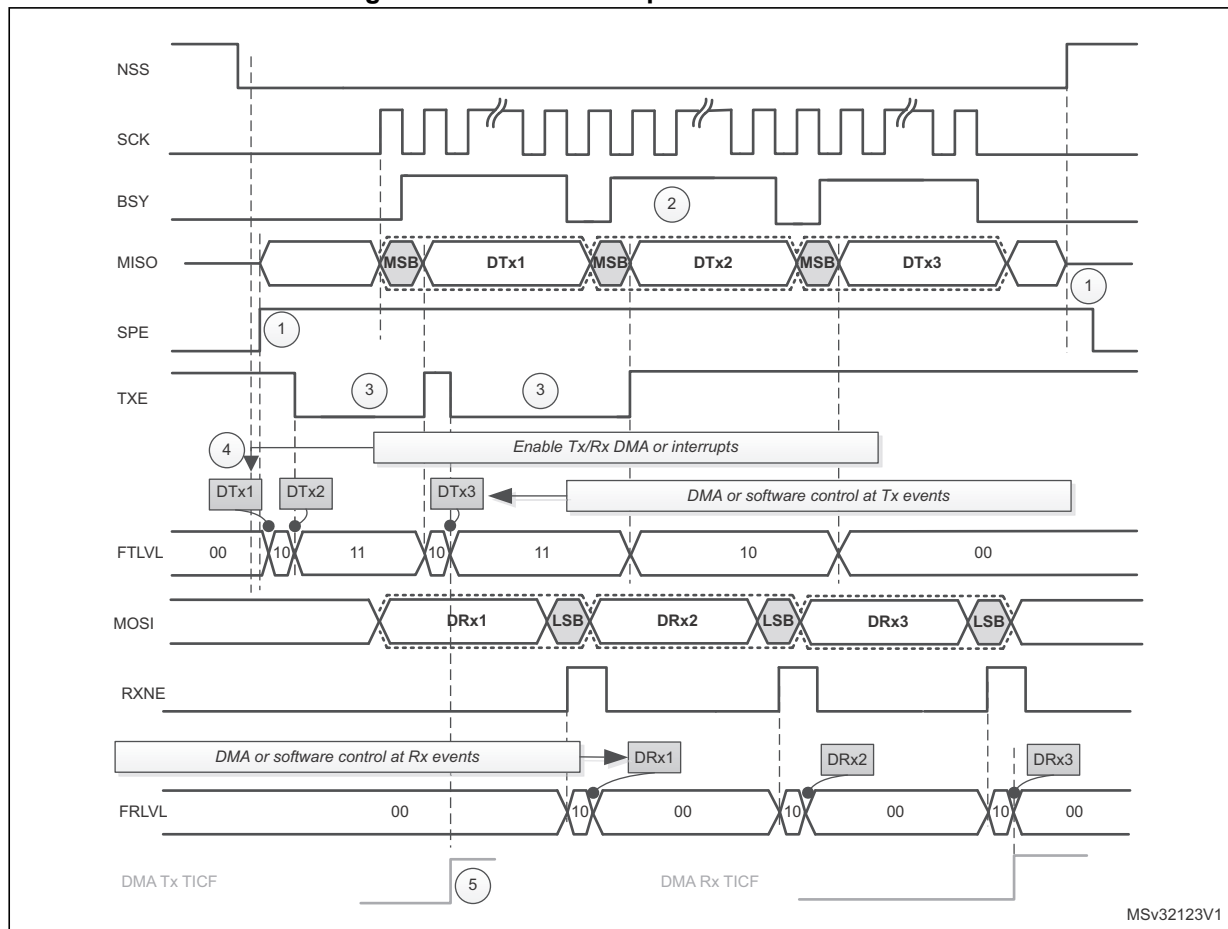
- Data size > 8 bit

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 918](#) for details about common assumptions and notes.

Figure 349. Slave full duplex communication



Assumptions for slave full duplex communication example:

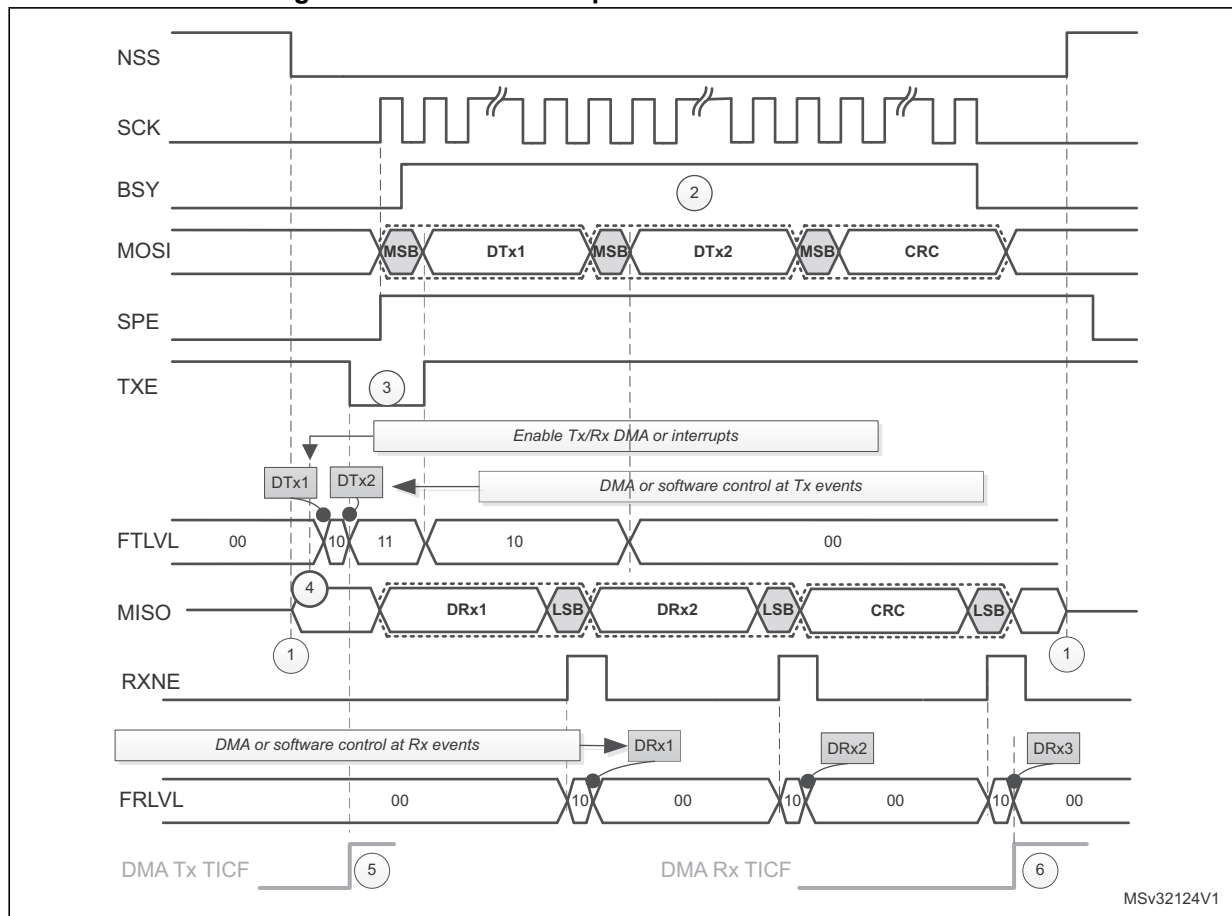
- Data size > 8 bit

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 918](#) for details about common assumptions and notes.

Figure 350. Master full duplex communication with CRC



Assumptions for master full duplex communication with CRC example:

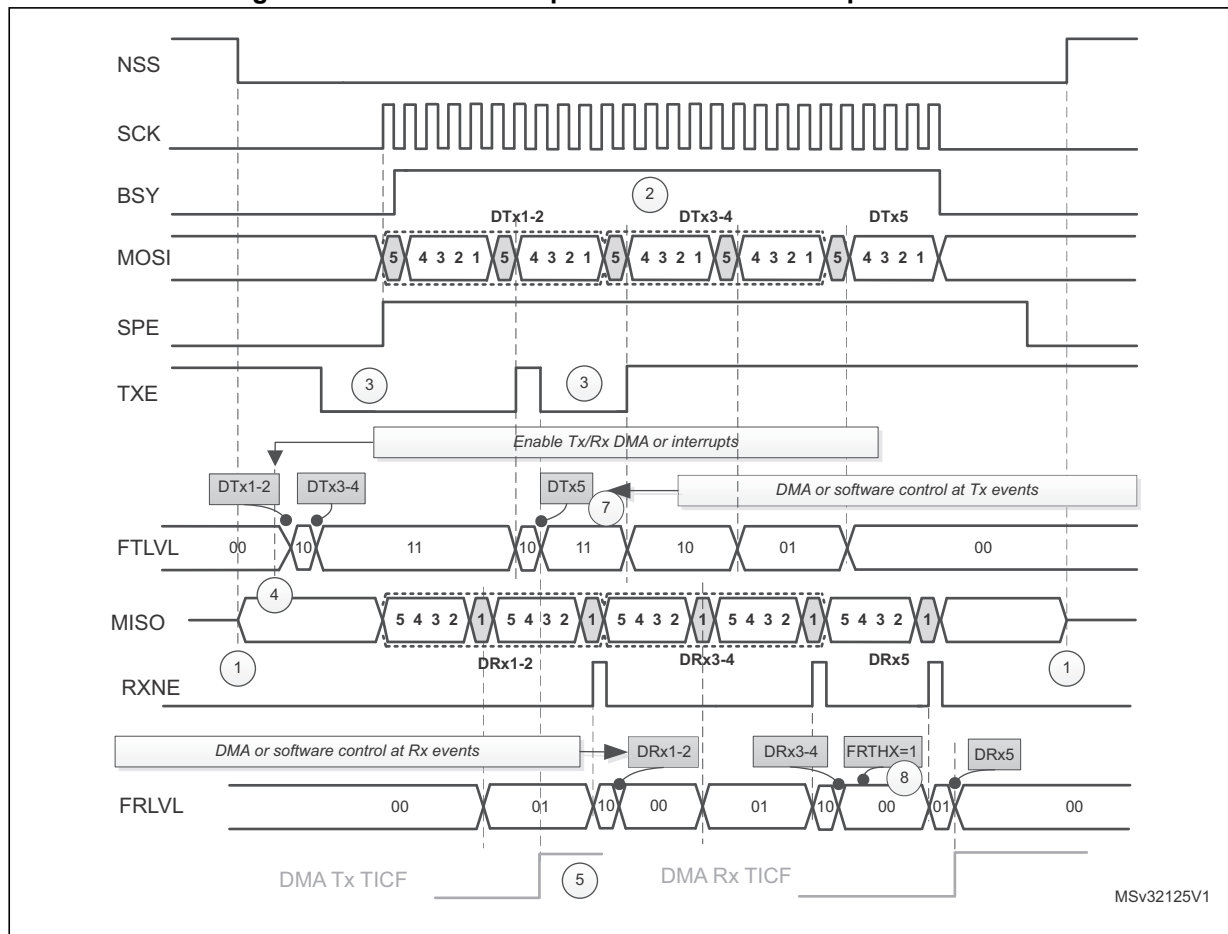
- Data size = 16 bit
- CRC enabled

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 918](#) for details about common assumptions and notes.

Figure 351. 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 918](#) for details about common assumptions and notes.

30.5.9 SPI status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

The TXE flag is set when transmission TXFIFO has enough space to store data to send. TXE flag is linked to the TXFIFO level. The flag goes high and stays high until the TXFIFO level is lower or equal to 1/2 of the FIFO depth. An interrupt can be generated if the TXEIE bit in the SPIx_CR2 register is set. The bit is cleared automatically when the TXFIFO level becomes greater than 1/2.

Rx buffer not empty (RXNE)

The RXNE flag is set depending on the FRXTH bit value in the SPIx_CR2 register:

- If FRXTH is set, RXNE goes high and stays high until the RXFIFO level is greater or equal to 1/4 (8-bit).
- If FRXTH is cleared, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/2 (16-bit).

An interrupt can be generated if the RXNEIE bit in the SPIx_CR2 register is set.

The RXNE is cleared by hardware automatically when the above conditions are no longer true.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect).

When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).

The BSY flag can be used in certain modes to detect the end of a transfer so that the software can disable the SPI or its peripheral clock before entering a low-power mode which does not provide a clock for the peripheral. This avoids corrupting the last transfer.

The BSY flag is also useful for preventing write collisions in a multimaster system.

The BSY flag is cleared under any one of the following conditions:

- When the SPI is correctly disabled
- When a fault is detected in Master mode (MODF bit set to 1)
- In Master mode, when it finishes a data transmission and no new data is ready to be sent
- In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

Note: When the next transmission can be handled immediately by the master (e.g. if the master is in Receive-only mode or its Transmit FIFO is not empty), communication is continuous and the BSY flag remains set to '1' between transfers on the master side. Although this is not the case with a slave, it is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.

30.5.10 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERRIE bit.

Overrun flag (OVR)

An overrun condition occurs when data is received by a master or slave and the RXFIFO has not enough space to store this received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RXFIFO) or when space for data storage is limited e.g. the RXFIFO is not available when CRC is enabled in receive only mode so in this case the reception buffer is limited into a single data frame buffer (see [Section 30.5.13: CRC calculation](#)).

When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO. The newly received value is discarded and all data transmitted subsequently is lost. Clearing the OVR bit is done by a read access to the SPI_DR register followed by a read access to the SPI_SR register.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit. Master mode fault affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPIx_SR register while the MODF bit is set.
2. Then write to the SPIx_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence. As a security, hardware does not allow the SPE and MSTR bits to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multimaster conflict.

CRC error (CRCERR)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx_CR1 register is set. The CRCERR flag in the SPIx_SR register is set if the value received in the shift register does not match the receiver SPIx_RXCRCR value. The flag is cleared by the software.

TI mode frame format error (FRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPIx_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of two data bytes.

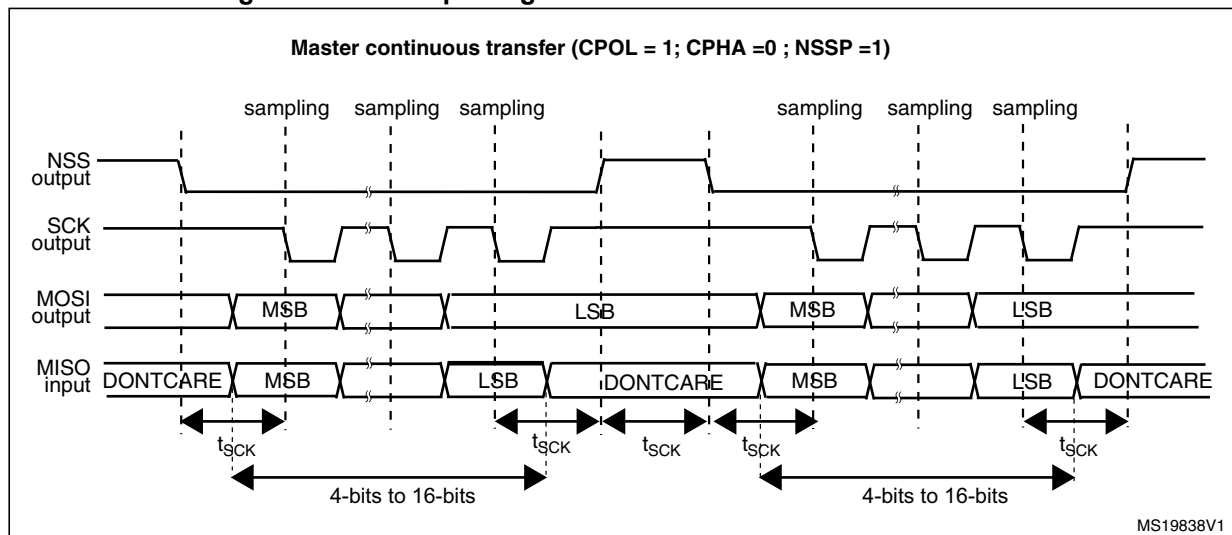
The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

30.5.11 NSS pulse mode

This mode is activated by the NSSP bit in the SPIx_CR2 register and it takes effect only if the SPI interface is configured as Motorola SPI master (FRF=0) with capture on the first edge (SPIx_CR1 CPHA = 0, CPOL setting is ignored). When activated, an NSS pulse is generated between two consecutive data frame transfers when NSS stays at high level for the duration of one clock period at least. This mode allows the slave to latch data. NSSP pulse mode is designed for applications with a single master-slave pair.

Figure 352 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 352. 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.

30.5.12 TI mode

TI protocol in master mode

The SPI interface is compatible with the TI protocol. The FRF bit of the SPIx_CR2 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPIx_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPIx_CR1 and SPIx_CR2 registers (SSM, SSI, SSOE) impossible in this case.

In slave mode, the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ when the current transaction finishes (see Figure 353). 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:

$$\frac{t_{\text{baud_rate}}}{2} + 4 \times t_{\text{pclk}} < t_{\text{release}} < \frac{t_{\text{baud_rate}}}{2} + 6 \times t_{\text{pclk}}$$

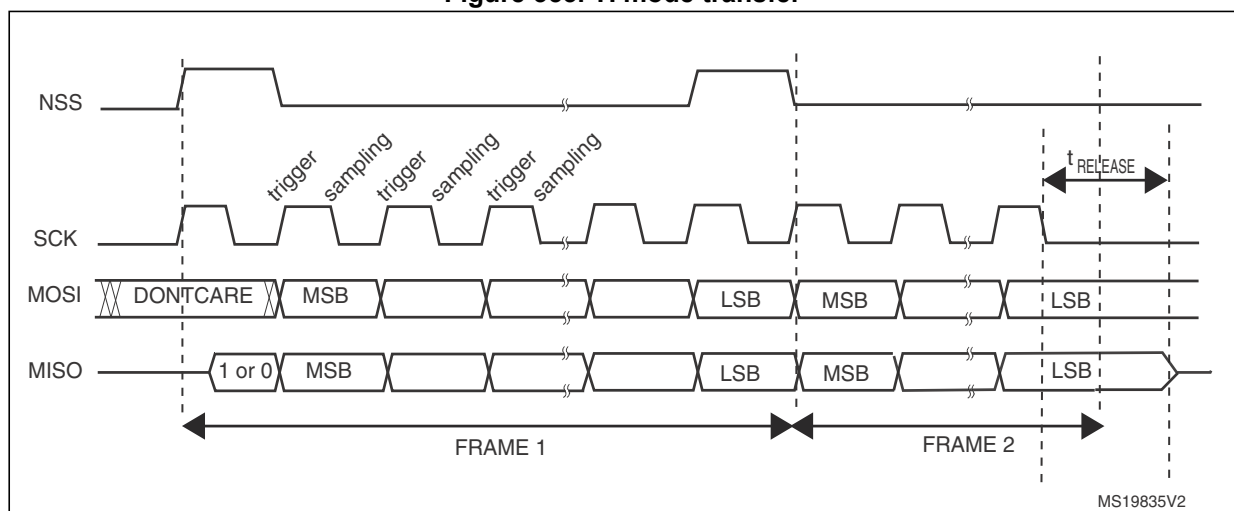
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

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 353: TI mode transfer shows the SPI communication waveforms when TI mode is selected.

Figure 353. TI mode transfer



30.5.13 CRC calculation

Two separate CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation independently of the frame data length, which can be fixed to 8-bit or 16-bit. For all the other data frame lengths, no CRC is available.

CRC principle

CRC calculation is enabled by setting the CRCEN bit in the SPIx_CR1 register before the SPI is enabled (SPE = 1). The CRC value is calculated using an odd programmable polynomial on each bit. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management.

Note: The polynomial value should only be odd. No even values are supported.

CRC transfer managed by CPU

Communication starts and continues normally until the last data frame has to be sent or received in the SPIx_DR register. Then CRCNEXT bit has to be set in the SPIx_CR1 register to indicate that the CRC frame transaction will follow after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the RXFIFO like a data byte or word. That is why in CRC mode only, the reception buffer has to be considered as a single 16-bit buffer used to receive only one data frame at a time.

A CRC-format transaction usually takes one more data frame to communicate at the end of data sequence. However, when setting an 8-bit data frame checked by 16-bit CRC, two more frames are necessary to send the complete CRC.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx_RXCRC register. Software has to check the CRCERR flag in the SPIx_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the RXFIFO and must be read in the SPIx_DR register in order to clear the RXNE flag.

CRC transfer managed by DMA

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out received CRC information from RXFIFO as it is always loaded into it. In full duplex mode, the counter of the reception DMA channel can be set to the number of data frames to receive including the CRC, which means, for example, in the specific case of an 8-bit data frame checked by 16-bit CRC:

$$\text{DMA_RX} = \text{Numb_of_data} + 2$$

In receive only mode, the DMA reception channel counter should contain only the amount of data transferred, excluding the CRC calculation. Then based on the complete transfer from DMA, all the CRC values must be read back by software from FIFO as it works as a single buffer in this mode.

At the end of the data and CRC transfers, the CRCERR flag in the SPIx_SR register is set if corruption occurred during the transfer.

If packing mode is used, the LDMA_RX bit needs managing if the number of data is odd.

Resetting the SPIx_TXCRC and SPIx_RXCRC values

The SPIx_TXCRC and SPIx_RXCRC values are cleared automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode (not available in receive-only mode) in order to transfer data without any interruption, (several data blocks covered by intermediate CRC checking phases).

If the SPI is disabled during a communication the following sequence must be followed:

1. Disable the SPI
2. Clear the CRCEN bit
3. Enable the CRCEN bit
4. Enable the SPI

Note: *When the SPI is in slave mode, the CRC calculator is sensitive to the SCK slave input clock as soon as the CRCEN bit is set, and this is the case whatever the value of the SPE bit. In order to avoid any wrong CRC calculation, the software must enable CRC calculation only when the clock is stable (in steady state). When the SPI interface is configured as a slave, the NSS internal signal needs to be kept low between the data phase and the CRC phase.*

30.6 SPI interrupts

During SPI communication an interrupts can be generated by the following events:

- Transmit TXFIFO ready to be loaded
- Data received in Receive RXFIFO
- Master mode fault
- Overrun error
- TI frame format error
- CRC protocol error

Interrupts can be enabled and disabled separately.

Table 163. SPI interrupt requests

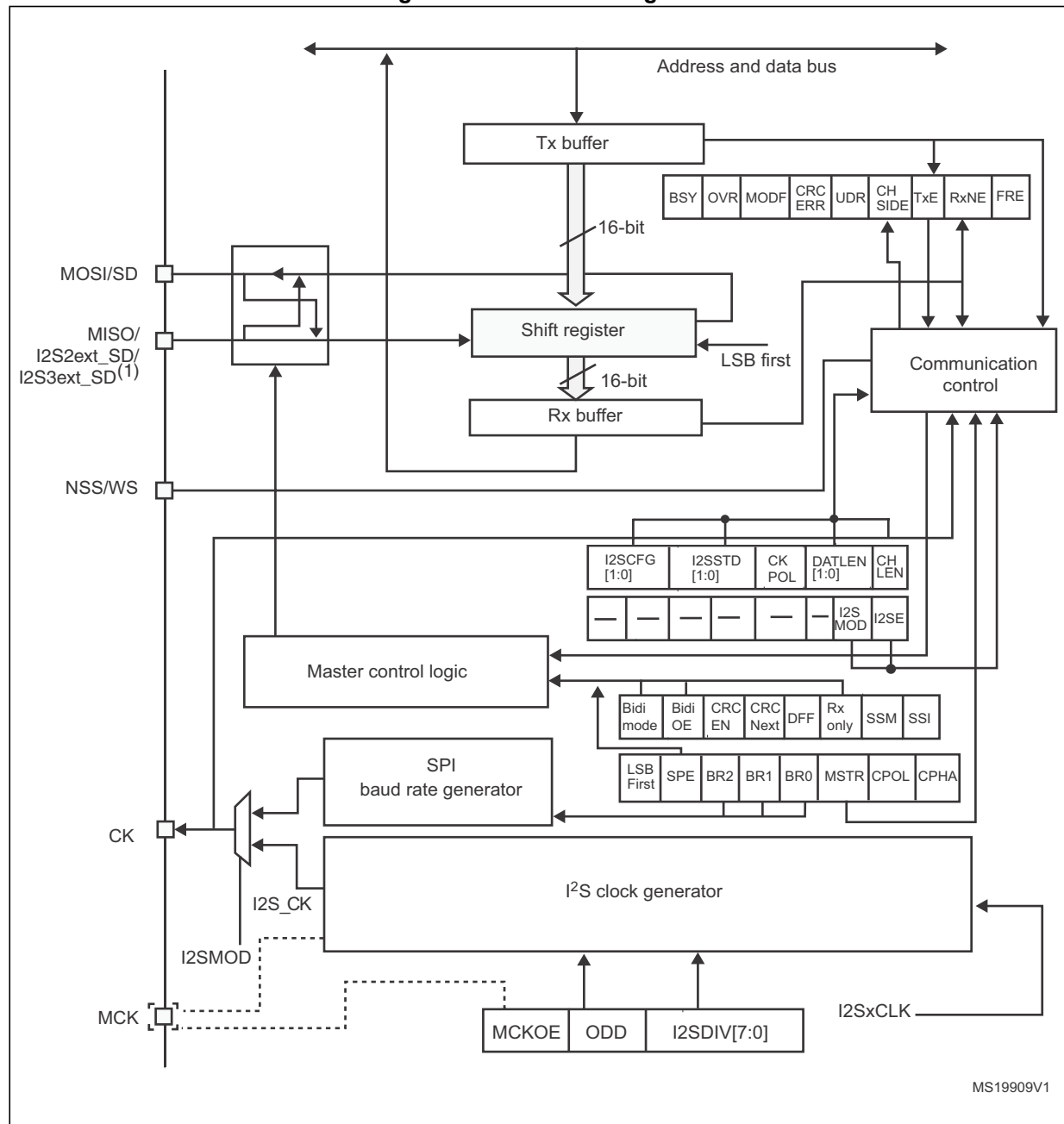
Interrupt event	Event flag	Enable Control bit
Transmit TXFIFO ready to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
TI frame format error	FRE	
CRC protocol error	CRCERR	

30.7 I²S functional description

30.7.1 I²S general description

The block diagram of the I²S is shown in [Figure 354](#).

Figure 354. I²S block diagram



1. I2S2ext_SD and I2S3ext_SD are the extended SD pins that control the I2S full duplex mode.

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.
- I2S2ext_SD and I2S3ext_SD: additional pins (mapped on the MISO pin) to control the I2S full duplex 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 $256 \times f_s$, 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 I²S 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 I²S mode. Likewise, the SSOE bit in the SPIx_CR2 register and the MODF and CRCERR bits in the SPIx_SR are not used.

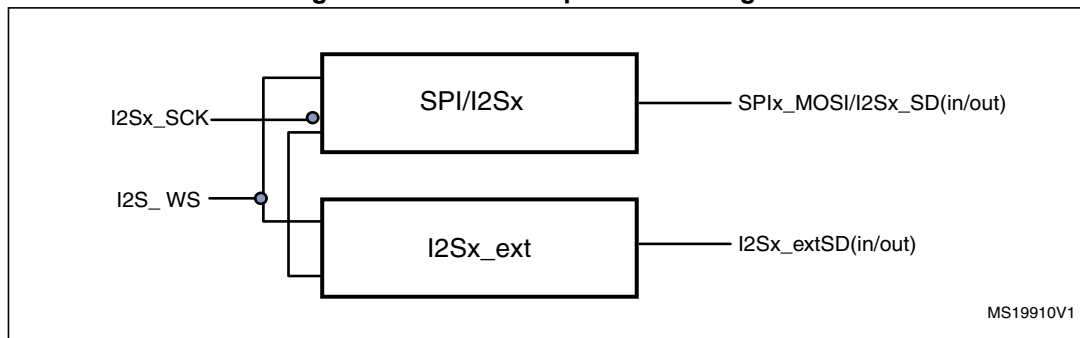
The I²S uses the same SPI register for data transfer (SPIx_DR) in 16-bit wide mode.

30.7.2 I2S full duplex

To support I2S full duplex mode, two extra I²S instances called extended I2Ss (I2S2_ext, I2S3_ext) are available in addition to I2S2 and I2S3 (see [Figure 355](#)). The first I2S full-duplex interface is consequently based on I2S2 and I2S2_ext, and the second one on I2S3 and I2S3_ext.

Note: I2S2_ext and I2S3_ext are used only in full-duplex mode.

Figure 355. I2S full duplex block diagram



1. Where x can be 2 or 3.

I2Sx can operate in master mode. As a result:

- Only I2Sx can output SCK and WS in half duplex mode
- Only I2Sx can deliver SCK and WS to I2S2_ext and I2S3_ext in full duplex mode.

The extended I2Ss (I2Sx_ext) can be used only in full duplex mode. The I2Sx_ext operate always in slave mode.

Both I2Sx and I2Sx_ext can be configured as transmitters or receivers.

30.7.3 Supported audio protocols

The four-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).

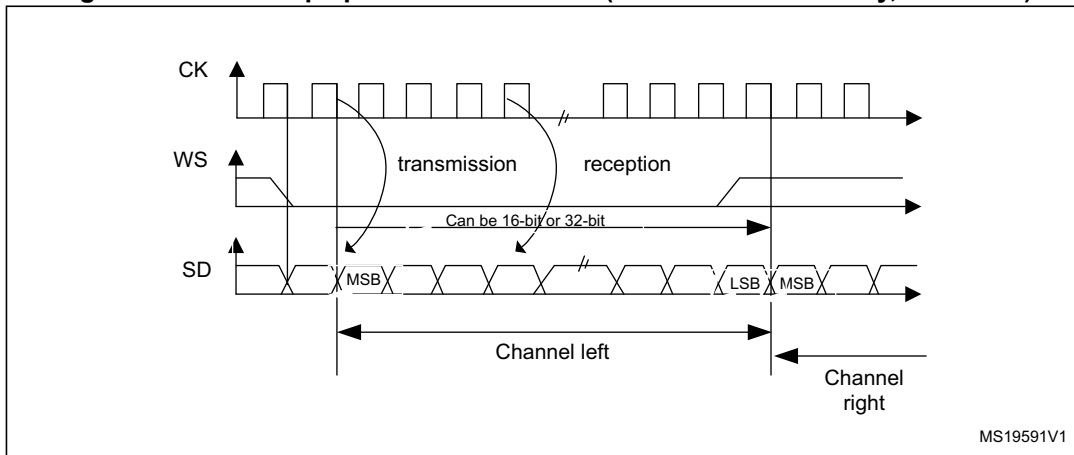
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

The I²S interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPIx_I2SCFGR register.

I²S Philips standard

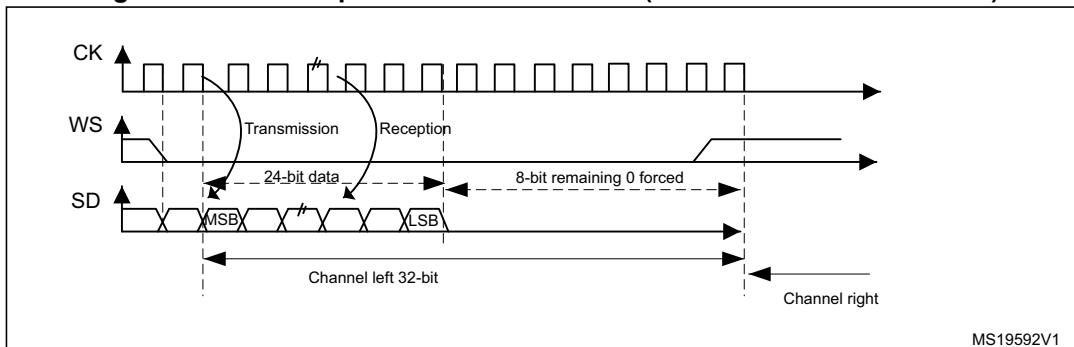
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

Figure 356. I²S Philips protocol waveforms (16/32-bit full accuracy, CPOL = 0)



Data are latched on the falling edge of CK (for the transmitter) and are read on the rising edge (for the receiver). The WS signal is also latched on the falling edge of CK.

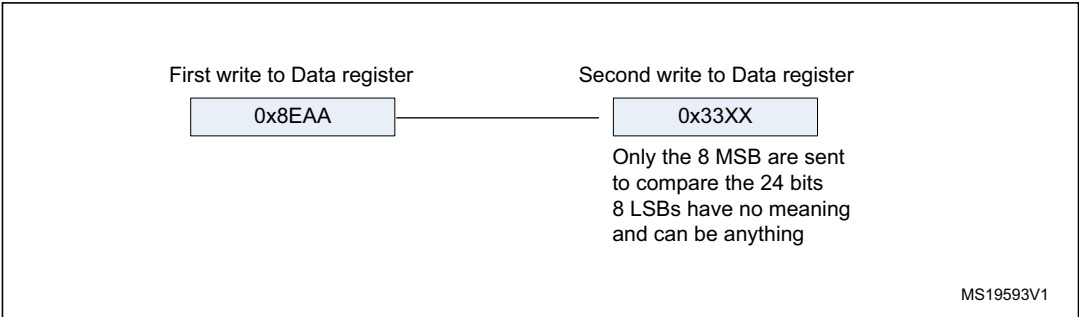
Figure 357. I²S Philips standard waveforms (24-bit frame with CPOL = 0)



This mode needs two write or read operations to/from the SPIx_DR register.

- In transmission mode:
If 0x8EAA33 has to be sent (24-bit):

Figure 358. Transmitting 0x8EAA33



- In reception mode:
If data 0x8EAA33 is received:

Figure 359. Receiving 0x8EAA33

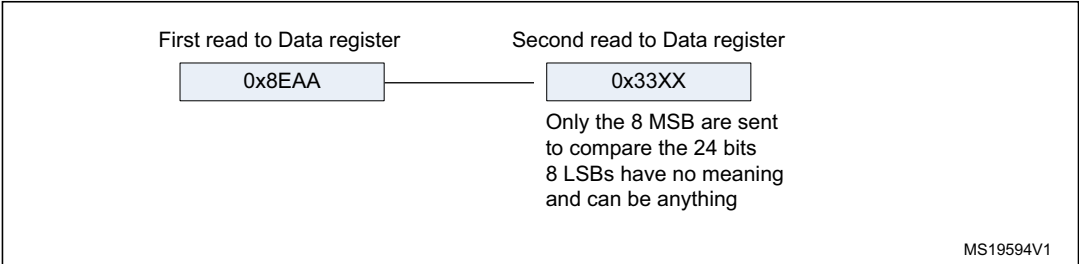
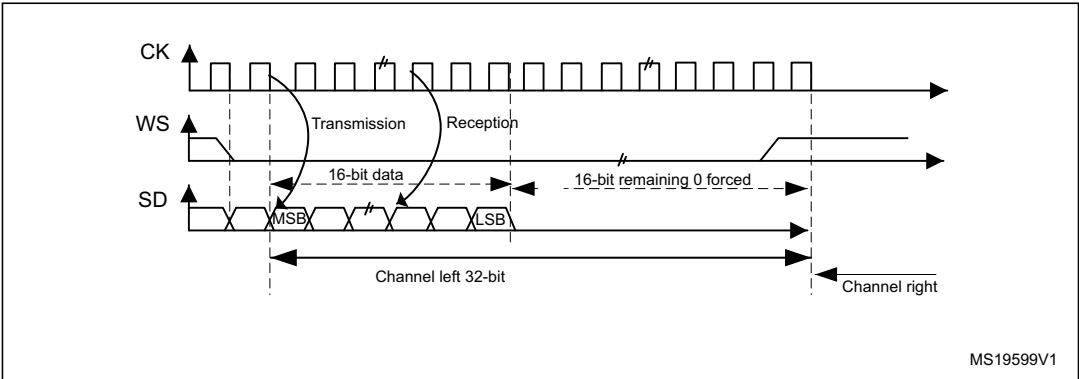


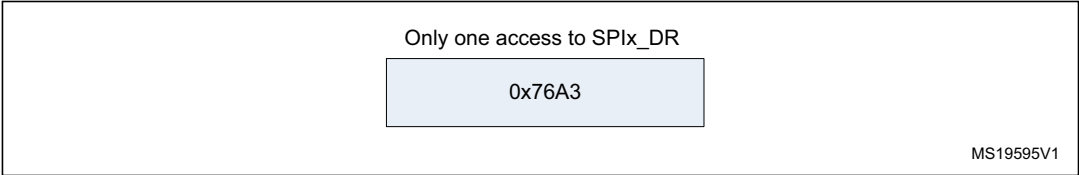
Figure 360. I²S Philips standard (16-bit extended to 32-bit packet frame with CPOL = 0)



When 16-bit data frame extended to 32-bit channel frame is selected during the 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 361](#) is required.

Figure 361. 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.

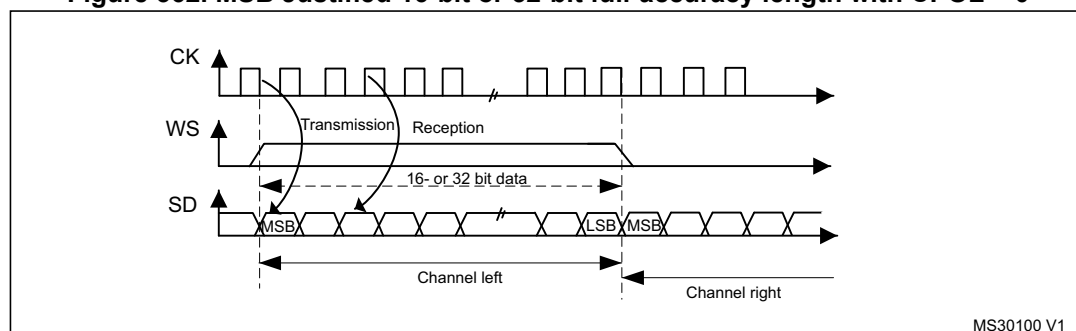
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

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 MSB.

Figure 362. MSB Justified 16-bit or 32-bit full-accuracy length with CPOL = 0



Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

Figure 363. MSB justified 24-bit frame length with CPOL = 0

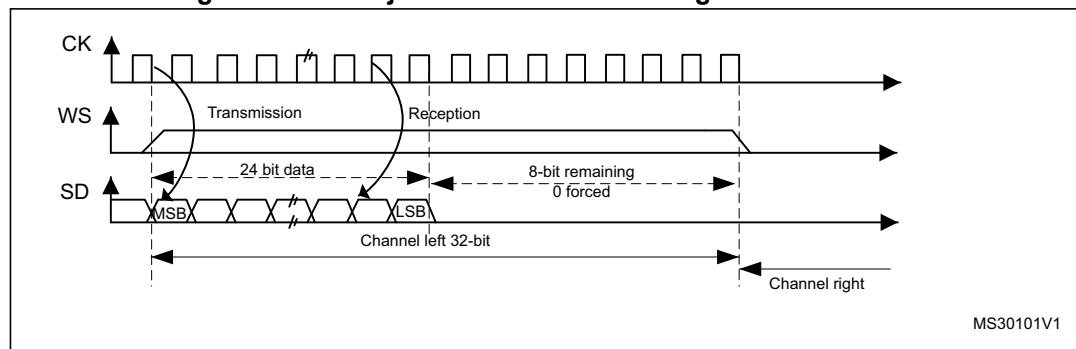
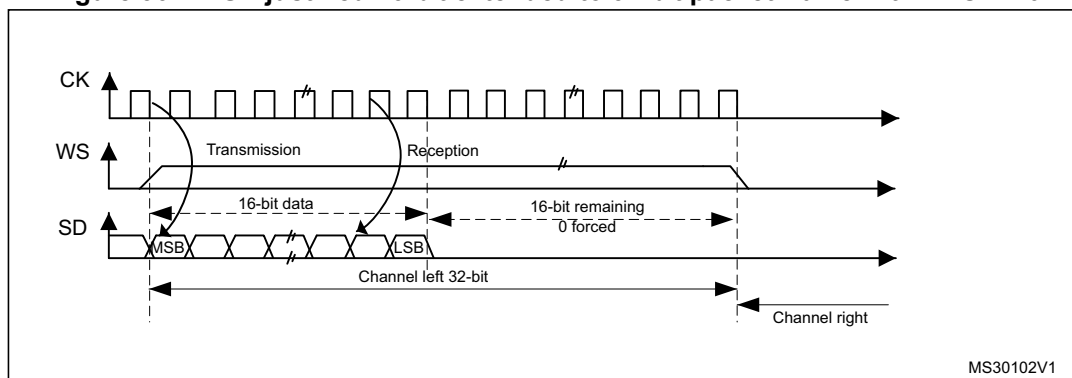
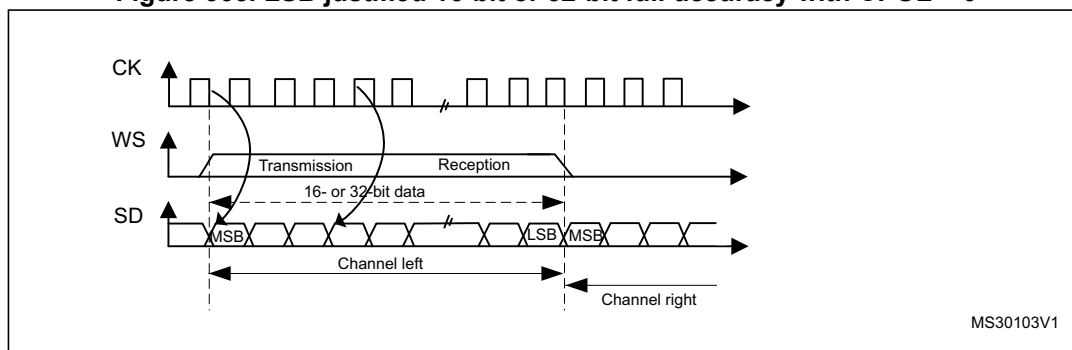
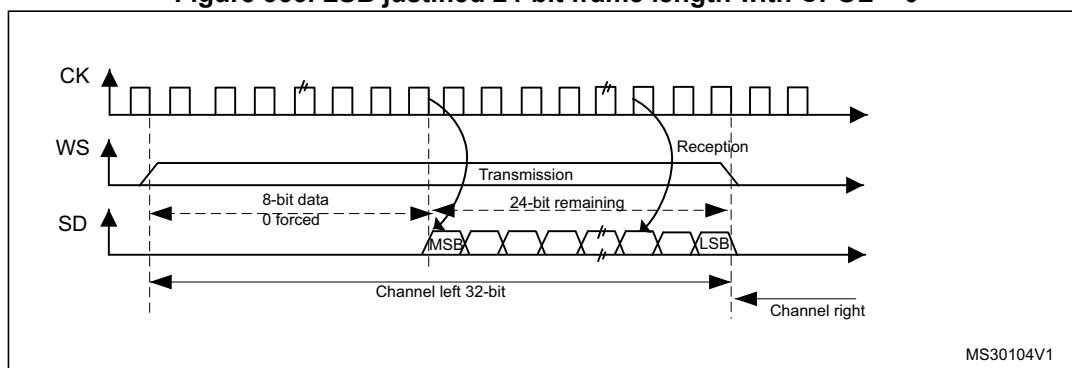
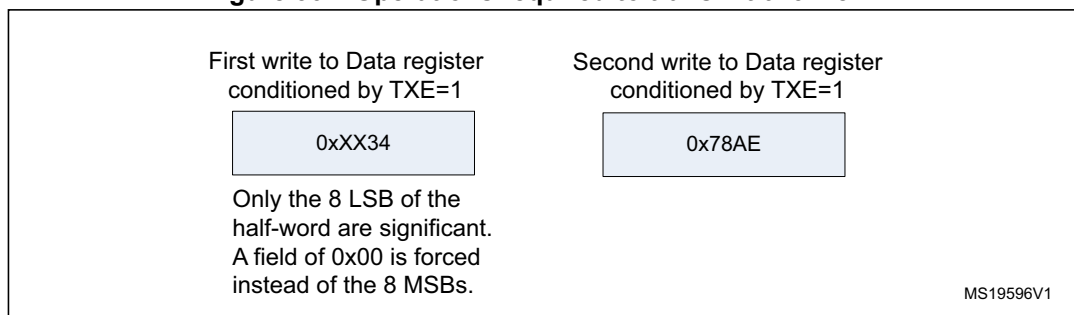


Figure 364. MSB justified 16-bit extended to 32-bit packet frame with CPOL = 0**LSB justified standard**

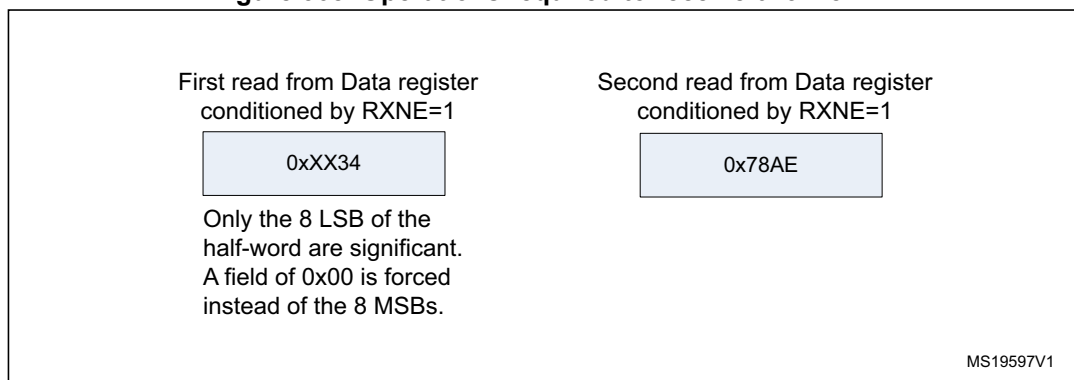
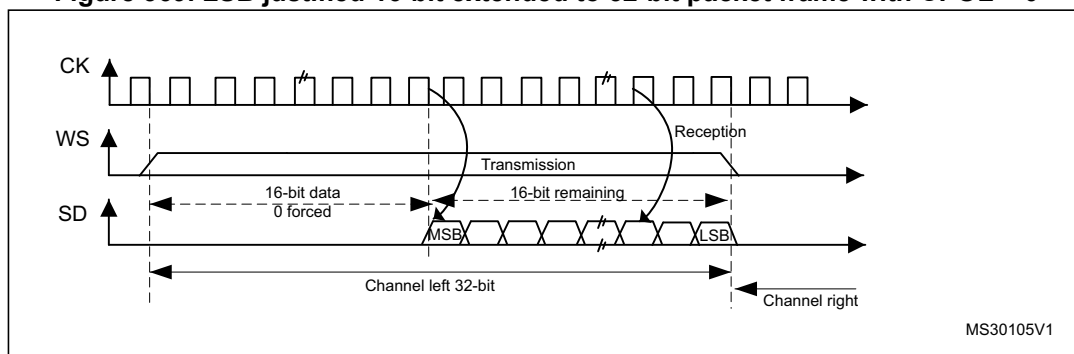
This standard is similar to the MSB justified standard (no difference for the 16-bit and 32-bit full-accuracy frame formats).

Figure 365. LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0**Figure 366. LSB justified 24-bit frame length with CPOL = 0**

- In transmission mode:
If data 0x3478AE have to be transmitted, two write operations to the SPIx_DR register are required by software or by DMA. The operations are shown below.

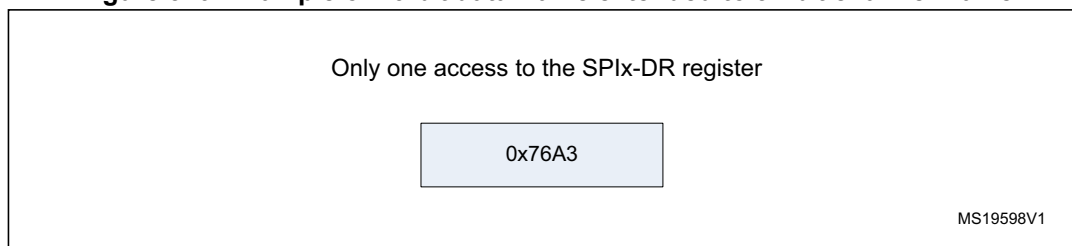
Figure 367. Operations required to transmit 0x3478AE

- In reception mode:
If data 0x3478AE are received, two successive read operations from the SPIx_DR register are required on each RXNE event.

Figure 368. Operations required to receive 0x3478AE**Figure 369. LSB justified 16-bit extended to 32-bit packet frame with CPOL = 0**

When 16-bit data frame extended to 32-bit channel frame is selected during the 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 370](#) is required.

Figure 370. Example of 16-bit data frame extended to 32-bit channel frame

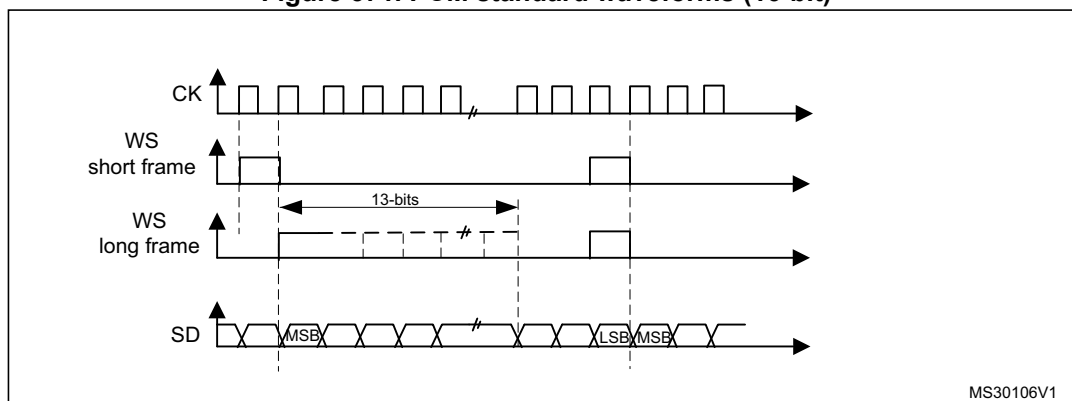
In transmission mode, when a TXE event occurs, the application has to write the data to be transmitted (in this case 0x76A3). The 0x000 field is transmitted first (extension on 32-bit). The TXE flag is set again as soon as the effective data (0x76A3) is sent on SD.

In reception mode, RXNE is asserted as soon as the significant half-word is received (and not the 0x0000 field).

In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

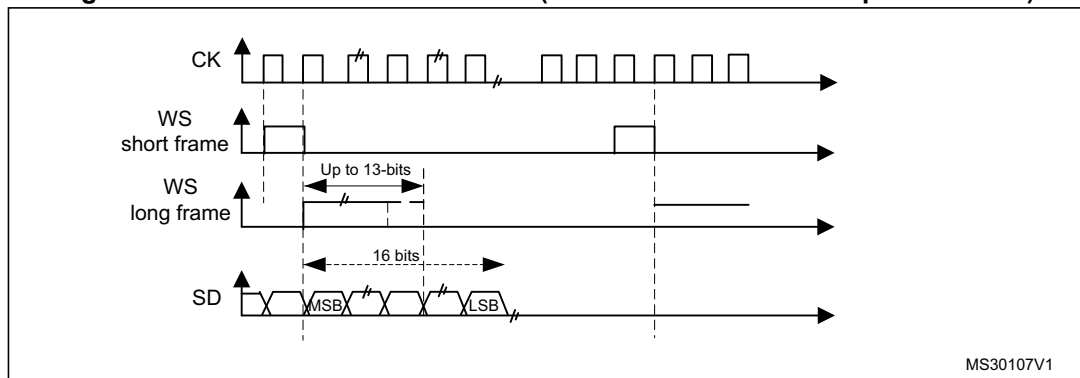
PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPIx_I2SCFGR register.

Figure 371. 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 372. PCM standard waveforms (16-bit extended to 32-bit packet frame)

Note: For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data (and so two synchronization signals) needs to be specified (DATLEN and CHLEN bits in the SPIx_I2SCFGR register) even in slave mode.

30.7.4 Clock generator

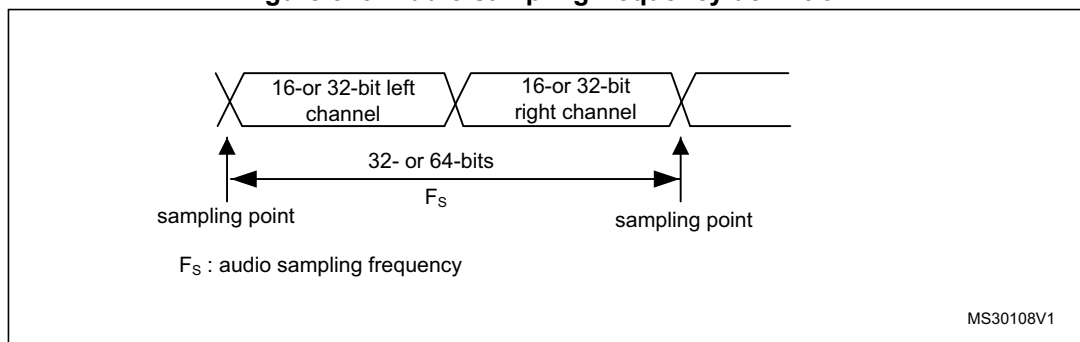
The I²S bit rate determines the data flow on the I²S data line and the I²S clock signal frequency.

I²S bit rate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I²S bit rate is calculated as follows:

$$\text{I}^2\text{S bit rate} = 16 \times 2 \times f_s$$

It will be: I²S bit rate = 32 × 2 × f_s if the packet length is 32-bit wide.

Figure 373. 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 374 presents the communication clock architecture. By default, the I2Sx clock is always the system clock. To achieve high-quality audio performance, the I2SxCLK clock source can be an external clock (mapped to the I2S_CKIN pin). Refer to [Section 10.4.2: Clock configuration register \(RCC_CFGR\)](#).

$$f_s = \text{ISxCLK} / [(32 \times 2) \times ((2 \times \text{ISDIV}) + \text{ODD}) \times 4]$$
 when the channel frame is 32-bit wide
$$f_s = I2SxCLK / [(32 \times 2) \times ((2 \times I2SDIV) + ODD)]$$
 when the channel frame is 32-bit wide

Note: Other configurations are possible that allow optimum clock precision.

SYSCLK (MHz)	I2S_DIV		I2S_ODD		MCLK	Target f _s (Hz)	Real f _s (KHz)		Error	
	16-bit	32-bit	16-bit	32-bit			16-bit	32-bit	16-bit	32-bit
72	11	6	1	0	No	96000	97826.09	93750	1.90%	2.34%
72	23	11	1	1	No	48000	47872.34	48913.04	0.27%	1.90%
72	25	13	1	0	No	44100	44117.65	43269.23	0.04%	1.88%
72	35	17	0	1	No	32000	32142.86	32142.86	0.44%	0.44%
72	51	25	0	1	No	22050	22058.82	22058.82	0.04%	0.04%
72	70	35	1	0	No	16000	15675.75	16071.43	0.27%	0.45%

Table 164. Audio-frequency precision using standard 8 MHz HSE⁽¹⁾ (continued)

SYSCLK (MHz)	I2S_DIV		I2S_ODD		MCLK	Target f _s (Hz)	Real f _s (KHz)		Error	
	16-bit	32-bit	16-bit	32-bit			16-bit	32-bit	16-bit	32-bit
72	102	51	0	0	No	11025	11029.41	11029.41	0.04%	0.04%
72	140	70	1	1	No	8000	8007.11	7978.72	0.09%	0.27%
72	3	3	0	0	Yes	48000	46875	46875	2.34%	2.34%
72	3	3	0	0	Yes	44100	46875	46875	6.29%	6.29%
72	9	9	0	0	Yes	32000	31250	31250	2.34%	2.34%
72	6	6	1	1	Yes	22050	21634.61	21634.61	1.88%	1.88%
72	9	9	0	0	Yes	16000	15625	15625	2.34%	2.34%
72	13	13	0	0	Yes	11025	10817.30	10817.30	1.88%	1.88%
72	17	17	1	1	Yes	8000	8035.71	8035.71	0.45%	0.45%

1. This table gives only example values for different clock configurations. Other configurations allowing optimum clock precision are possible.

30.7.5 I²S master mode

The I²S can be configured as follows:

- In master mode for transmission or reception (half-duplex mode using I2Sx)
- In master mode transmission and reception (full duplex mode using I2Sx and I2Sx_ext).

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 30.7.4: Clock generator](#)).
3. Set the I2SMOD bit in the SPIx_I2SCFGR register to activate the I²S functions and choose the I²S 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 I²S 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 I²S standard mode selected, refer to [Section 30.7.3: 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 30.7.5: I2S master mode](#)), where the configuration should set the master reception mode through the I2SCFG[1:0] bits.

Whatever the data or channel length, the audio data are received by 16-bit packets. This means that each time the Rx buffer is full, the RXNE flag is set and an interrupt is generated if the RXNEIE bit is set in SPIx_CR2 register. Depending on the data and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the Rx buffer.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

CHSIDE is updated after each reception. It is sensitive to the WS signal generated by the I²S cell.

For more details about the read operations depending on the I²S standard mode selected, refer to [Section 30.7.3: 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, I²S 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.

30.7.6 I²S slave mode

The I2S can be configured as follows:

- In slave mode for transmission or reception (half duplex mode using I2Sx)
- In slave mode transmission and reception (full duplex mode using I2Sx and I2Sx_ext).

The operating mode is following mainly the same rules as described for the I²S 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 I²S mode and choose the I²S 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 (see note below).

Note: The I2S slave must be enabled after the external master sets the WS line at high level if the I2S protocol is selected, or at low level if the LSB or MSB-justified mode is selected.

Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the NSS_WS signal requests the transfer of data. The slave has to be enabled before

the external master starts the communication. The 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 I²S standard mode selected, refer to [Section 30.7.3: 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 30.7.6: I2S slave mode](#)), where the configuration should set the master reception mode using the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.

Whatever the data length or the channel length, the audio data are received by 16-bit packets. This means that each time the RX buffer is full, the RXNE flag in the SPIx_SR register is set and an interrupt is generated if the RXNEIE bit is set in the SPIx_CR2 register. Depending on the data length and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the RX buffer.

The CHSIDE flag is updated each time data are received to be read from the SPIx_DR register. It is sensitive to the external WS line managed by the external master component.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

For more details about the read operations depending on the I²S standard mode selected, refer to [Section 30.7.3: 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.

30.7.7 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).

30.7.8 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 I²S mode or low for MSB- or LSB-justified or PCM modes).

Desynchronization between master and slave devices may be due to noisy environment on the SCK communication clock or on the WS frame synchronization line. An error interrupt can be generated if the ERRIE bit is set. The desynchronization flag (FRE) is cleared by software when the status register is read.

30.7.9 DMA features

In I²S 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 I²S mode since there is no data transfer protection system.

30.8 I²S interrupts

Table 165 provides the list of I²S interrupts.

Table 165. 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	

30.9 SPI and I²S registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI_DR in addition by can be accessed by 8-bit access.

30.9.1 SPI control register 1 (SPIx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
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 I²S 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 I²S mode.

Bit 13 **CRCCEN**: Hardware CRC calculation enable

0: CRC calculation disabled

1: CRC calculation Enabled

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation.

This bit is not used in I²S 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 I²S 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 I²S 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 I²S 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 I²S 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 I²S 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 I²S 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 914](#).

This bit is not used in I²S mode.

Bits 5:3 **BR[2:0]**: Baud rate control

000: $f_{PCLK}/2$

001: $f_{PCLK}/4$

010: $f_{PCLK}/8$

011: $f_{PCLK}/16$

100: $f_{PCLK}/32$

101: $f_{PCLK}/64$

110: $f_{PCLK}/128$

111: $f_{PCLK}/256$

Note: These bits should not be changed when communication is ongoing.

This bit is not used in I²S 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 I²S mode.

Bit1 **CPOL**: Clock polarity

0: CK to 0 when idle

1: CK to 1 when idle

Note: This bit should not be changed when communication is ongoing.

This bit is not used in SPI TI mode.

Bit 0 **CPHA**: Clock phase

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

Note: This bit should not be changed when communication is ongoing.

This bit is not used in SPI TI mode.

30.9.2 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 914](#) if the CRCEN bit is set.

This bit is not used in I²S 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 914](#) if the CRCEN bit is set.

This bit is not used in I²S 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 I²S mode.

Bit 11:8 **DS [3:0]**: Data size

These bits configure the data length for SPI transfers:

0000: Not used
 0001: Not used
 0010: Not used
 0011: 4-bit
 0100: 5-bit
 0101: 6-bit
 0110: 7-bit
 0111: 8-bit
 1000: 9-bit
 1001: 10-bit
 1010: 11-bit
 1011: 12-bit
 1100: 13-bit
 1101: 14-bit
 1110: 15-bit
 1111: 16-bit

If software attempts to write one of the “Not used” values, they are forced to the value “0111”(8-bit).

Note: This bit is not used in I²S 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).

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 I²S 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 SPI TI mode.

Bit 2 **SSOE**: SS output enable

0: SS output is disabled in master mode and the SPI interface can work in multimaster configuration

1: SS output is enabled in master mode and when the SPI interface is enabled. The SPI interface cannot work in a multimaster environment.

Note: This bit is not used in SPI TI mode.

Bit 1 **TXDMAEN**: Tx buffer DMA enable

When this bit is set, a DMA request is generated whenever the TXE flag is set.

0: Tx buffer DMA disabled

1: Tx buffer DMA enabled

Bit 0 **RXDMAEN**: Rx buffer DMA enable

When this bit is set, a DMA request is generated whenever the RXNE flag is set.

0: Rx buffer DMA disabled

1: Rx buffer DMA enabled

30.9.3 SPI status register (SPIx_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTLVL[1:0]		FRLVL[2:0]		FRE	BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
			r	r	r	r	r	r	r	r	rc_w0	r	r	r	r

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:11 **FTLVL[1:0]**: FIFO Transmission Level

These bits are set and cleared by hardware.

00: FIFO empty

01: 1/4 FIFO

10: 1/2 FIFO

11: FIFO full (considered as FULL when the FIFO threshold is greater than 1/2)

Note: These bits are not used in I²S 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 I²S 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 I²S slave mode. Refer to [Section 30.5.10: SPI error flags](#) and [Section 30.7.8: 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 30.5.9: SPI status flags and Procedure for disabling the SPI on page 914](#).

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 945](#) 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 924](#) for the software sequence.

Note: This bit is not used in I²S mode.

Bit 4 **CRCERR**: CRC error flag

0: CRC value received matches the SPIx_RXCRCR value

1: CRC value received does not match the SPIx_RXCRCR value

This flag is set by hardware and cleared by software writing 0.

*Note: This bit is not used in I²S 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 945](#) 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

30.9.4 SPI data register (SPIx_DR)

Address offset: 0x0C

Reset value: 0x0000

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

Bits 15:0 **DR[15:0]**: Data register

Data received or to be transmitted

The data register serves as an interface between the Rx and Tx FIFOs. When the data register is read, RxFIFO is accessed while the write to data register accesses TxFIFO (See [Section 30.5.8: Data transmission and reception procedures](#)).

Note: Data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read. The Rx threshold setting must always correspond with the read access currently used.

30.9.5 SPI CRC polynomial register (SPIx_CRCPR)

Address offset: 0x10

Reset value: 0x0007

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

Bits 15:0 **CRCPOLY[15:0]**: CRC polynomial register

This register contains the polynomial for the CRC calculation.

The CRC polynomial (0007h) is the reset value of this register. Another polynomial can be configured as required.

Note: The polynomial value should be odd only. No even value is supported.

30.9.6 SPI Rx CRC register (SPIx_RXCRCR)

Address offset: 0x14

Reset value: 0x0000

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

Bits 15:0 **RxCRC[15:0]**: Rx CRC register

When CRC calculation is enabled, the RxCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPIx_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY Flag is set could return an incorrect value.

These bits are not used in I²S mode.

30.9.7 SPI Tx CRC register (SPIx_TXCRCR)

Address offset: 0x18

Reset value: 0x0000

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

Bits 15:0 **TxCRC[15:0]**: Tx CRC register

When CRC calculation is enabled, the TxCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPIx_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (CRCL bit in the SPIx_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (CRCL bit in the SPIx_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY flag is set could return an incorrect value.

These bits are not used in I²S mode.

30.9.8 SPIx_I2S configuration register (SPIx_I2SCFGR)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
				rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved: Forced to 0 by hardware

Bit 11 **I2SMOD**: I2S mode selection

0: SPI mode is selected

1: I2S mode is selected

Note: This bit should be configured when the SPI is disabled.

Bit 10 **I2SE**: I2S enable

0: I²S peripheral is disabled

1: I²S peripheral is enabled

Note: This bit is not used in SPI mode.

Bits 9:8 **I2SCFG**: I2S configuration mode

00: Slave - transmit

01: Slave - receive

10: Master - transmit

11: Master - receive

Note: These bits should be configured when the I²S is disabled.

They are not used in SPI mode.

Bit 7 **PCMSYNC**: PCM frame synchronization

0: Short frame synchronization

1: Long frame synchronization

Note: This bit has a meaning only if I2SSTD = 11 (PCM standard is used).

It is not used in SPI mode.

Bit 6 Reserved: forced at 0 by hardware

Bits 5:4 **I2SSTD**: I²S standard selection

00: I²S Philips standard.

01: MSB justified standard (left justified)

10: LSB justified standard (right justified)

11: PCM standard

For more details on I²S standards, refer to [Section 30.7.3 on page 931](#)

Note: For correct operation, these bits should be configured when the I²S is disabled.

They are not used in SPI mode.

Bit 3 **CKPOL**: Steady state clock polarity

0: I²S clock steady state is low level

1: I²S clock steady 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.

Bits 2:1 **DATLEN**: Data length to be transferred

00: 16-bit data length

01: 24-bit data length

10: 32-bit data length

11: Not allowed

Note: For correct operation, these bits should be configured when the 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.

30.9.9 SPIx_I2S prescaler register (SPIx_I2SPR)

Address offset: 0x20

Reset value: 0000 0010 (0x0002)

Bits 15:10 Reserved: Forced to 0 by hardware

Bit 9 **MCKOE**: Master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Note: This bit should be configured when the 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 30.7.4 on page 938](#)

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**: I²S linear prescaler

I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values.

Refer to [Section 30.7.4 on page 938](#)

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.

30.9.10 SPI/I2S register map

Table 166 shows the SPI/I2S register map and reset values.

Table 166. SPI register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0x00	SPIx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	0	0				
0x08	SPIx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTLV[1:0]	FRLV[1:0]		FRE		BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE						
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	1	0				
0x0C	SPIx_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[15:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	SPIx_CRCPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCPOLY[15:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1			
0x14	SPIx_RXCRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RxCRC[15:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x18	SPIx_TXCRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2SMOD	I2SE	I2SCFG	PCMSYNC		Res.		I2SSTD	CKPOL		DATLEN	CHLEN							
	Reset value																		0	0	0	0	0	0		0	0	0	0	0	0	0	0				
0x20	SPIx_I2SPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV														
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	1	0				

Refer to [Section 3.2.2 on page 48](#) for the register boundary addresses.

31 Controller area network (bxCAN)

31.1 Introduction

The **Basic Extended CAN** peripheral, named **bxCAN**, interfaces the CAN network. It supports the CAN protocols version 2.0A and B. It has been designed to manage a high number of incoming messages efficiently with a minimum CPU load. It also meets the priority requirements for transmit messages.

For safety-critical applications, the CAN controller provides all hardware functions for supporting the CAN Time Triggered Communication option.

31.2 bxCAN main features

- Supports CAN protocol version 2.0 A, B Active
- Bit rates up to 1 Mbit/s
- Supports the Time Triggered Communication option

Transmission

- Three transmit mailboxes
- Configurable transmit priority
- Time Stamp on SOF transmission

Reception

- Two receive FIFOs with three stages
- Scalable filter banks:
 - 14 filter banks
- Identifier list feature
- Configurable FIFO overrun
- Time Stamp on SOF reception

Time-triggered communication option

- Disable automatic retransmission mode
- 16-bit free running timer
- Time Stamp sent in last two data bytes

Management

- Maskable interrupts
- Software-efficient mailbox mapping at a unique address space

31.3 bxCAN general description

In today's CAN applications, the number of nodes in a network is increasing and often several networks are linked together via gateways. Typically the number of messages in the system (and thus to be handled by each node) has significantly increased. In addition to the application messages, Network Management and Diagnostic messages have been introduced.

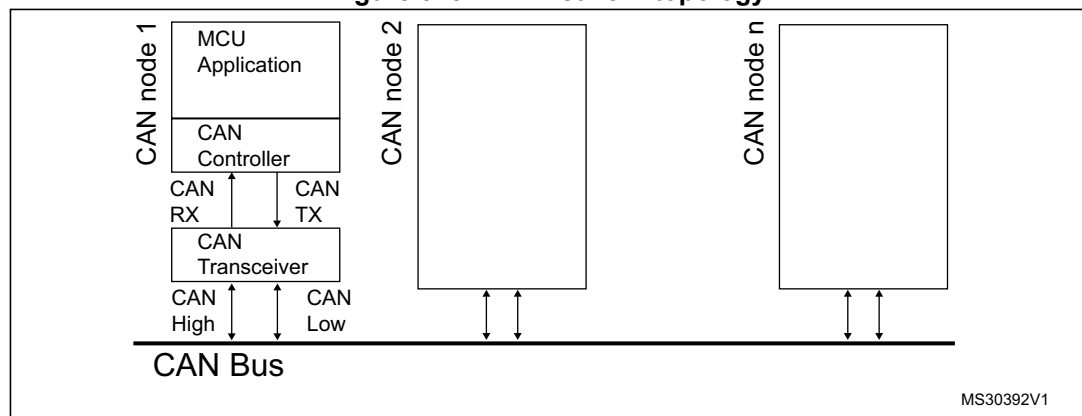
- An enhanced filtering mechanism is required to handle each type of message.

Furthermore, application tasks require more CPU time, therefore real-time constraints caused by message reception have to be reduced.

- A receive FIFO scheme allows the CPU to be dedicated to application tasks for a long time period without losing messages.

The standard HLP (Higher Layer Protocol) based on standard CAN drivers requires an efficient interface to the CAN controller.

Figure 375. CAN network topology



31.3.1 CAN 2.0B active core

The bxCAN module handles the transmission and the reception of CAN messages fully autonomously. Standard identifiers (11-bit) and extended identifiers (29-bit) are fully supported by hardware.

31.3.2 Control, status and configuration registers

The application uses these registers to:

- Configure CAN parameters, e.g. baud rate
- Request transmissions
- Handle receptions
- Manage interrupts
- Get diagnostic information

31.3.3 Tx mailboxes

Three transmit mailboxes are provided to the software for setting up messages. The transmission Scheduler decides which mailbox has to be transmitted first.

31.3.4 Acceptance filters

The bxCAN provides 14 scalable/configurable identifier filter banks for selecting the incoming messages the software needs and discarding the others.

Receive FIFO

Two receive FIFOs are used by hardware to store the incoming messages. Three complete messages can be stored in each FIFO. The FIFOs are managed completely by hardware.

31.4 bxCAN operating modes

bxCAN has three main operating modes: **initialization**, **normal** and **Sleep**. After a hardware reset, bxCAN is in Sleep mode to reduce power consumption and an internal pull-up is active on CANTX. The software requests bxCAN to enter **initialization** or **Sleep** mode by setting the INRQ or SLEEP bits in the CAN_MCR register. Once the mode has been entered, bxCAN confirms it by setting the INAK or SLAK bits in the CAN_MSR register and the internal pull-up is disabled. When neither INAK nor SLAK are set, bxCAN is in **normal** mode. Before entering **normal** mode bxCAN always has to **synchronize** on the CAN bus. To synchronize, bxCAN waits until the CAN bus is idle, this means 11 consecutive recessive bits have been monitored on CANRX.

31.4.1 Initialization mode

The software initialization can be done while the hardware is in Initialization mode. To enter this mode the software sets the INRQ bit in the CAN_MCR register and waits until the hardware has confirmed the request by setting the INAK bit in the CAN_MSR register.

To leave Initialization mode, the software clears the INQR bit. bxCAN has left Initialization mode once the INAK bit has been cleared by hardware.

While in Initialization Mode, all message transfers to and from the CAN bus are stopped and the status of the CAN bus output CANTX is recessive (high).

Entering Initialization Mode does not change any of the configuration registers.

To initialize the CAN Controller, software has to set up the Bit Timing (CAN_BTR) and CAN options (CAN_MCR) registers.

To initialize the registers associated with the CAN filter banks (mode, scale, FIFO assignment, activation and filter values), software has to set the FINIT bit (CAN_FMR). Filter initialization also can be done outside the initialization mode.

Note: When FINIT=1, CAN reception is deactivated.

The filter values also can be modified by deactivating the associated filter activation bits (in the CAN_FA1R register).

If a filter bank is not used, it is recommended to leave it non active (leave the corresponding FACT bit cleared).

31.4.2 Normal mode

Once the initialization is complete, the software must request the hardware to enter Normal mode to be able to synchronize on the CAN bus and start reception and transmission.

The request to enter Normal mode is issued by clearing the INRQ bit in the CAN_MCR register. The bxCAN enters Normal mode and is ready to take part in bus activities when it has synchronized with the data transfer on the CAN bus. This is done by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus Idle state). The switch to Normal mode is confirmed by the hardware by clearing the INAK bit in the CAN_MSR register.

The initialization of the filter values is independent from Initialization Mode but must be done while the filter is not active (corresponding FACTx bit cleared). The filter scale and mode configuration must be configured before entering Normal Mode.

31.4.3 Sleep mode (low-power)

To reduce power consumption, bxCAN has a low-power mode called Sleep mode. This mode is entered on software request by setting the SLEEP bit in the CAN_MCR register. In this mode, the bxCAN clock is stopped, however software can still access the bxCAN mailboxes.

If software requests entry to **initialization** mode by setting the INRQ bit while bxCAN is in **Sleep** mode, it must also clear the SLEEP bit.

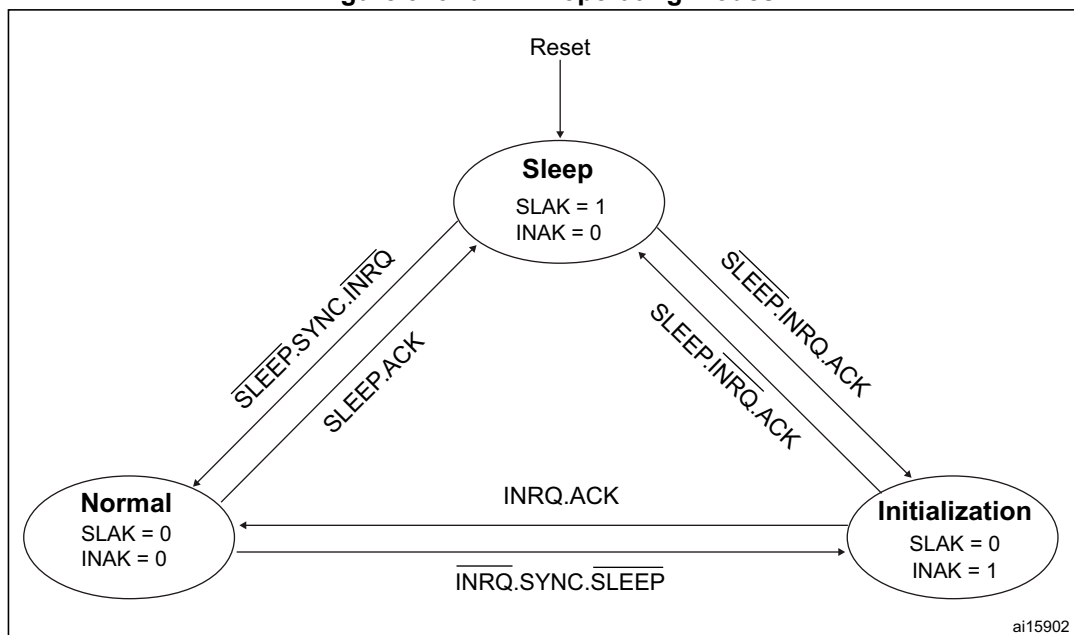
bxCAN can be woken up (exit Sleep mode) either by software clearing the SLEEP bit or on detection of CAN bus activity.

On CAN bus activity detection, hardware automatically performs the wakeup sequence by clearing the SLEEP bit if the AWUM bit in the CAN_MCR register is set. If the AWUM bit is cleared, software has to clear the SLEEP bit when a wakeup interrupt occurs, in order to exit from Sleep mode.

Note: If the wakeup interrupt is enabled (WKUIE bit set in CAN_IER register) a wakeup interrupt will be generated on detection of CAN bus activity, even if the bxCAN automatically performs the wakeup sequence.

After the SLEEP bit has been cleared, Sleep mode is exited once bxCAN has synchronized with the CAN bus, refer to [Figure 376: bxCAN operating modes](#). The Sleep mode is exited once the SLAK bit has been cleared by hardware.

Figure 376. bxCAN operating modes



1. ACK = The wait state during which hardware confirms a request by setting the INAK or SLAK bits in the CAN_MSR register
2. SYNC = The state during which bxCAN waits until the CAN bus is idle, meaning 11 consecutive recessive bits have been monitored on CANRX

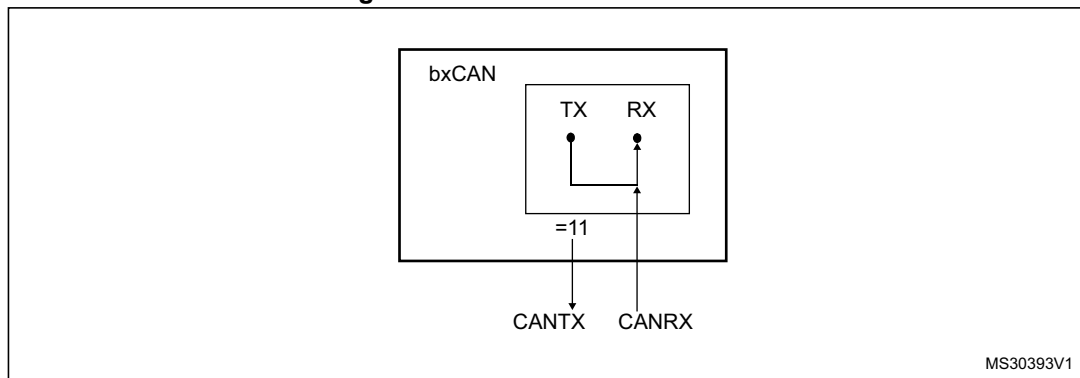
31.5 Test mode

Test mode can be selected by the SILM and LBKM bits in the CAN_BTR register. These bits must be configured while bxCAN is in Initialization mode. Once test mode has been selected, the INRQ bit in the CAN_MCR register must be reset to enter Normal mode.

31.5.1 Silent mode

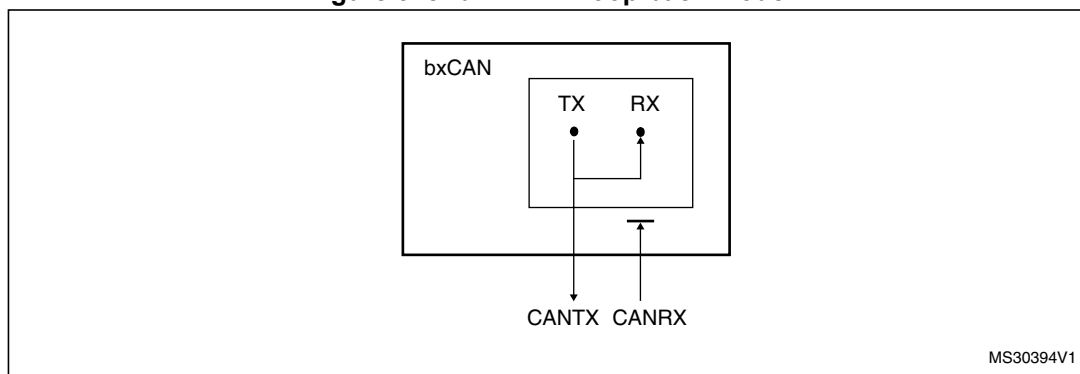
The bxCAN can be put in Silent mode by setting the SILM bit in the CAN_BTR register.

In Silent mode, the bxCAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the bxCAN has to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN Core monitors this dominant bit, although the CAN bus may remain in recessive state. Silent mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames).

Figure 377. bxCAN in silent mode

31.5.2 Loop back mode

The bxCAN can be set in Loop Back Mode by setting the LBKM bit in the CAN_BTR register. In Loop Back Mode, the bxCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) in a Receive mailbox.

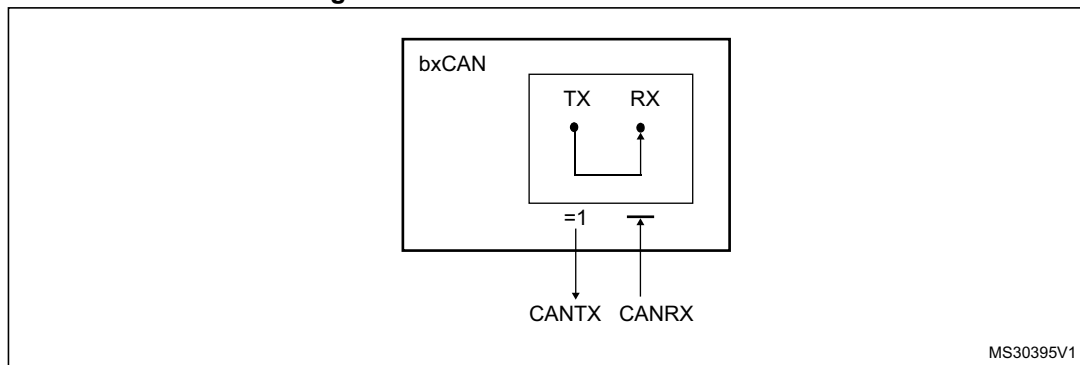
Figure 378. bxCAN in loop back mode

This mode is provided for self-test functions. To be independent of external events, the CAN Core ignores acknowledge errors (no dominant bit sampled in the acknowledge slot of a data / remote frame) in Loop Back Mode. In this mode, the bxCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CANRX input pin is disregarded by the bxCAN. The transmitted messages can be monitored on the CANTX pin.

31.5.3 Loop back combined with silent mode

It is also possible to combine Loop Back mode and Silent mode by setting the LBKM and SILM bits in the CAN_BTR register. This mode can be used for a “Hot Selftest”, meaning the bxCAN can be tested like in Loop Back mode but without affecting a running CAN system connected to the CANTX and CANRX pins. In this mode, the CANRX pin is disconnected from the bxCAN and the CANTX pin is held recessive.

Figure 379. bxCAN in combined mode



31.6 Behavior in Debug mode

When the microcontroller enters the debug mode (Cortex-M4[®]F core halted), the bxCAN continues to work normally or stops, depending on:

- the DBF bit in CAN_MCR. For more details, refer to [Section 31.9.2: CAN control and status registers](#).

31.7 bxCAN functional description

31.7.1 Transmission handling

In order to transmit a message, the application must select one **empty** transmit mailbox, set up the identifier, the data length code (DLC) and the data before requesting the transmission by setting the corresponding TXRQ bit in the CAN_TlRxR register. Once the mailbox has left **empty** state, the software no longer has write access to the mailbox registers. Immediately after the TXRQ bit has been set, the mailbox enters **pending** state and waits to become the highest priority mailbox, see *Transmit Priority*. As soon as the mailbox has the highest priority it will be **scheduled** for transmission. The transmission of the message of the scheduled mailbox will start (enter **transmit** state) when the CAN bus becomes idle. Once the mailbox has been successfully transmitted, it will become **empty** again. The hardware indicates a successful transmission by setting the RQCP and TXOK bits in the CAN_TSR register.

If the transmission fails, the cause is indicated by the ALST bit in the CAN_TSR register in case of an Arbitration Lost, and/or the TERR bit, in case of transmission error detection.

Transmit priority

By identifier

When more than one transmit mailbox is pending, the transmission order is given by the identifier of the message stored in the mailbox. The message with the lowest identifier value has the highest priority according to the arbitration of the CAN protocol. If the identifier values are equal, the lower mailbox number will be scheduled first.

By transmit request order

The transmit mailboxes can be configured as a transmit FIFO by setting the TXFP bit in the CAN_MCR register. In this mode the priority order is given by the transmit request order.

This mode is very useful for segmented transmission.

Abort

A transmission request can be aborted by the user setting the ABRQ bit in the CAN_TSR register. In **pending** or **scheduled** state, the mailbox is aborted immediately. An abort request while the mailbox is in **transmit** state can have two results. If the mailbox is transmitted successfully the mailbox becomes **empty** with the TXOK bit set in the CAN_TSR register. If the transmission fails, the mailbox becomes **scheduled**, the transmission is aborted and becomes **empty** with TXOK cleared. In all cases the mailbox will become **empty** again at least at the end of the current transmission.

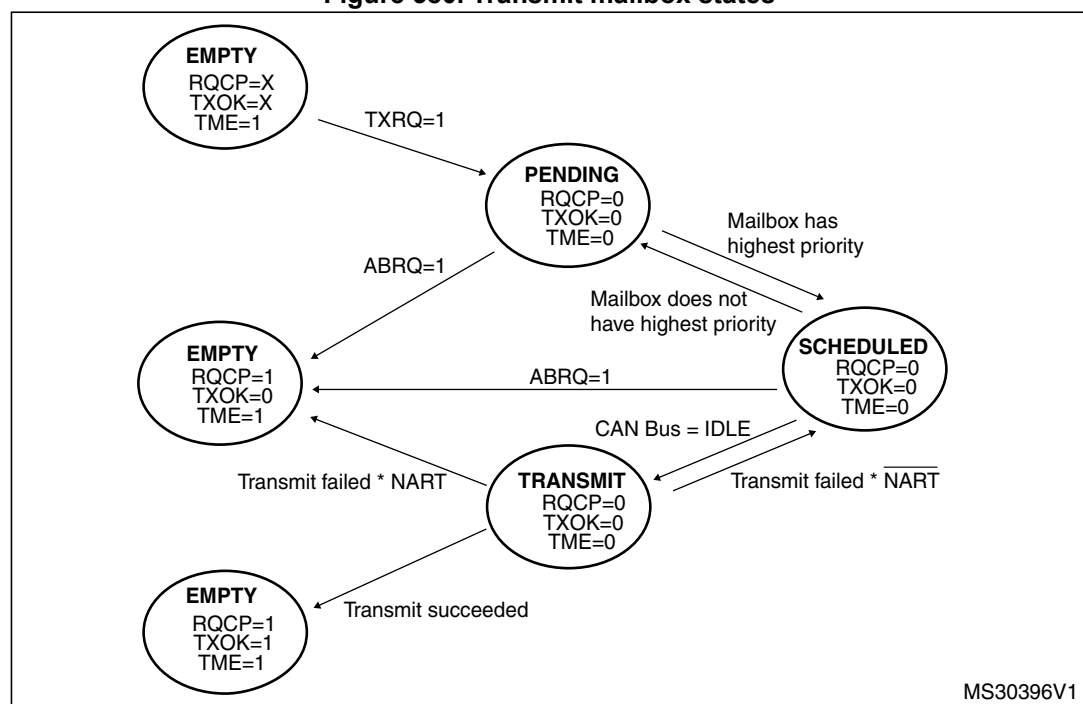
Non automatic retransmission mode

This mode has been implemented in order to fulfill the requirement of the Time Triggered Communication option of the CAN standard. To configure the hardware in this mode the NART bit in the CAN_MCR register must be set.

In this mode, each transmission is started only once. If the first attempt fails, due to an arbitration loss or an error, the hardware will not automatically restart the message transmission.

At the end of the first transmission attempt, the hardware considers the request as completed and sets the RQCP bit in the CAN_TSR register. The result of the transmission is indicated in the CAN_TSR register by the TXOK, ALST and TERR bits.

Figure 380. Transmit mailbox states



31.7.2 Time triggered communication mode

In this mode, the internal counter of the CAN hardware is activated and used to generate the Time Stamp value stored in the CAN_RDTxR/CAN_TDTxR registers, respectively (for Rx and Tx mailboxes). The internal counter is incremented each CAN bit time (refer to [Section 31.7.7: Bit timing](#)). The internal counter is captured on the sample point of the Start Of Frame bit in both reception and transmission.

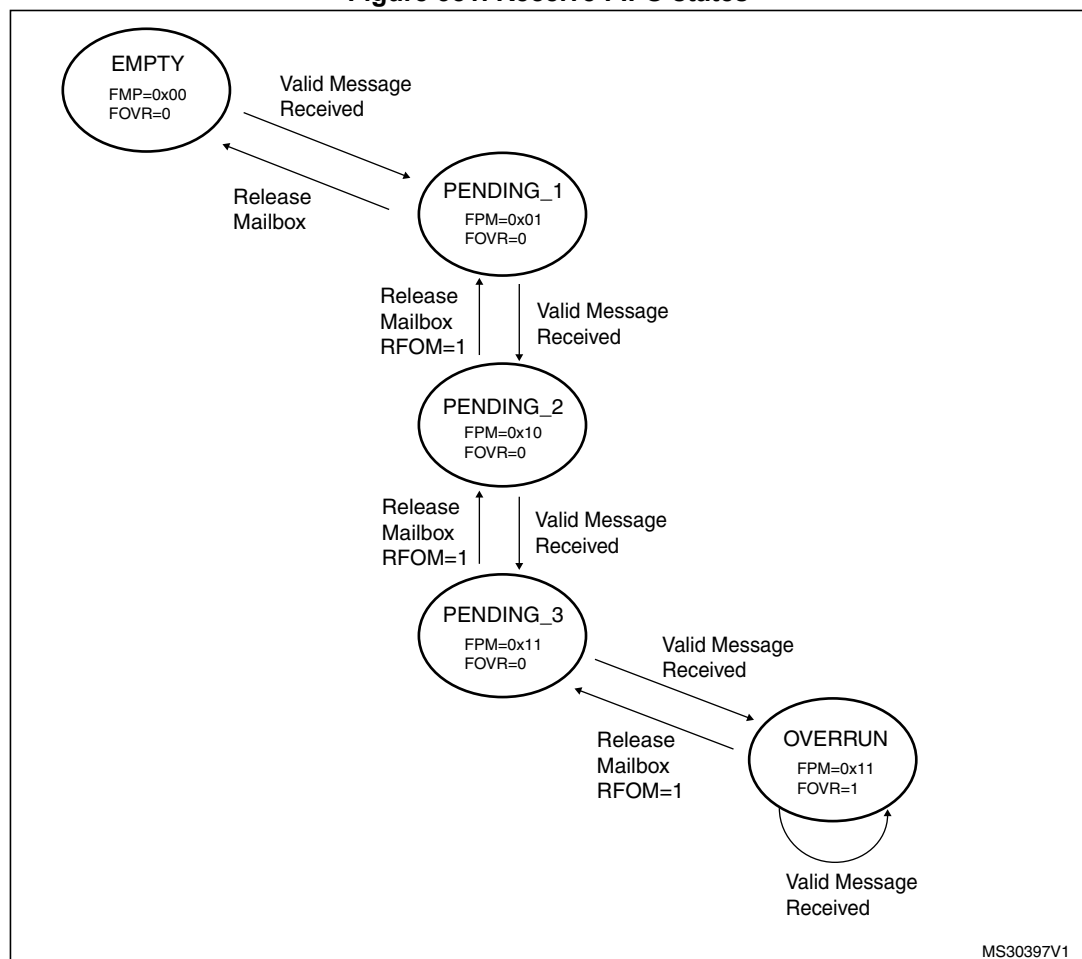
31.7.3 Reception handling

For the reception of CAN messages, three mailboxes organized as a FIFO are provided. In order to save CPU load, simplify the software and guarantee data consistency, the FIFO is managed completely by hardware. The application accesses the messages stored in the FIFO through the FIFO output mailbox.

Valid message

A received message is considered as valid **when** it has been received correctly according to the CAN protocol (no error until the last but one bit of the EOF field) **and** it passed through the identifier filtering successfully, see [Section 31.7.4: Identifier filtering](#).

Figure 381. Receive FIFO states



FIFO management

Starting from the **empty** state, the first valid message received is stored in the FIFO which becomes **pending_1**. The hardware signals the event setting the FMP[1:0] bits in the CAN_RFR register to the value 01b. The message is available in the FIFO output mailbox. The software reads out the mailbox content and releases it by setting the RFOM bit in the CAN_RFR register. The FIFO becomes **empty** again. If a new valid message has been received in the meantime, the FIFO stays in **pending_1** state and the new message is available in the output mailbox.

If the application does not release the mailbox, the next valid message will be stored in the FIFO which enters **pending_2** state (FMP[1:0] = 10b). The storage process is repeated for the next valid message putting the FIFO into **pending_3** state (FMP[1:0] = 11b). At this point, the software must release the output mailbox by setting the RFOM bit, so that a mailbox is free to store the next valid message. Otherwise the next valid message received will cause a loss of message.

Refer also to [Section 31.7.5: Message storage](#)

Overrun

Once the FIFO is in **pending_3** state (i.e. the three mailboxes are full) the next valid message reception will lead to an **overrun** and a message will be lost. The hardware signals the overrun condition by setting the FOVR bit in the CAN_RFR register. Which message is lost depends on the configuration of the FIFO:

- If the FIFO lock function is disabled (RFLM bit in the CAN_MCR register cleared) the last message stored in the FIFO will be overwritten by the new incoming message. In this case the latest messages will be always available to the application.
- If the FIFO lock function is enabled (RFLM bit in the CAN_MCR register set) the most recent message will be discarded and the software will have the three oldest messages in the FIFO available.

Reception related interrupts

Once a message has been stored in the FIFO, the FMP[1:0] bits are updated and an interrupt request is generated if the FMPIE bit in the CAN_IER register is set.

When the FIFO becomes full (i.e. a third message is stored) the FULL bit in the CAN_RFR register is set and an interrupt is generated if the FFIE bit in the CAN_IER register is set.

On overrun condition, the FOVR bit is set and an interrupt is generated if the FOVIE bit in the CAN_IER register is set.

31.7.4 Identifier filtering

In the CAN protocol the identifier of a message is not associated with the address of a node but related to the content of the message. Consequently a transmitter broadcasts its message to all receivers. On message reception a receiver node decides - depending on the identifier value - whether the software needs the message or not. If the message is needed, it is copied into the SRAM. If not, the message must be discarded without intervention by the software.

To fulfill this requirement, the bxCAN Controller provides 28 configurable and scalable filter banks (27-0) to the application. In other devices the bxCAN Controller provides 14

configurable and scalable filter banks (13-0) to the application in order to receive only the messages the software needs. This hardware filtering saves CPU resources which would be otherwise needed to perform filtering by software. Each filter bank x consists of two 32-bit registers, CAN_FxR0 and CAN_FxR1.

Scalable width

To optimize and adapt the filters to the application needs, each filter bank can be scaled independently. Depending on the filter scale a filter bank provides:

- One 32-bit filter for the STDID[10:0], EXTID[17:0], IDE and RTR bits.
- Two 16-bit filters for the STDID[10:0], RTR, IDE and EXTID[17:15] bits.

Refer to [Figure 382](#).

Furthermore, the filters can be configured in mask mode or in identifier list mode.

Mask mode

In **mask** mode the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don’t care”.

Identifier list mode

In **identifier list** mode, the mask registers are used as identifier registers. Thus instead of defining an identifier and a mask, two identifiers are specified, doubling the number of single identifiers. All bits of the incoming identifier must match the bits specified in the filter registers.

Filter bank scale and mode configuration

The filter banks are configured by means of the corresponding CAN_FMR register. To configure a filter bank it must be deactivated by clearing the FACT bit in the CAN_FAR register. The filter scale is configured by means of the corresponding FSCx bit in the CAN_FS1R register, refer to [Figure 382](#). The **identifier list** or **identifier mask** mode for the corresponding Mask/Identifier registers is configured by means of the FBMx bits in the CAN_FMR register.

To filter a group of identifiers, configure the Mask/Identifier registers in mask mode.

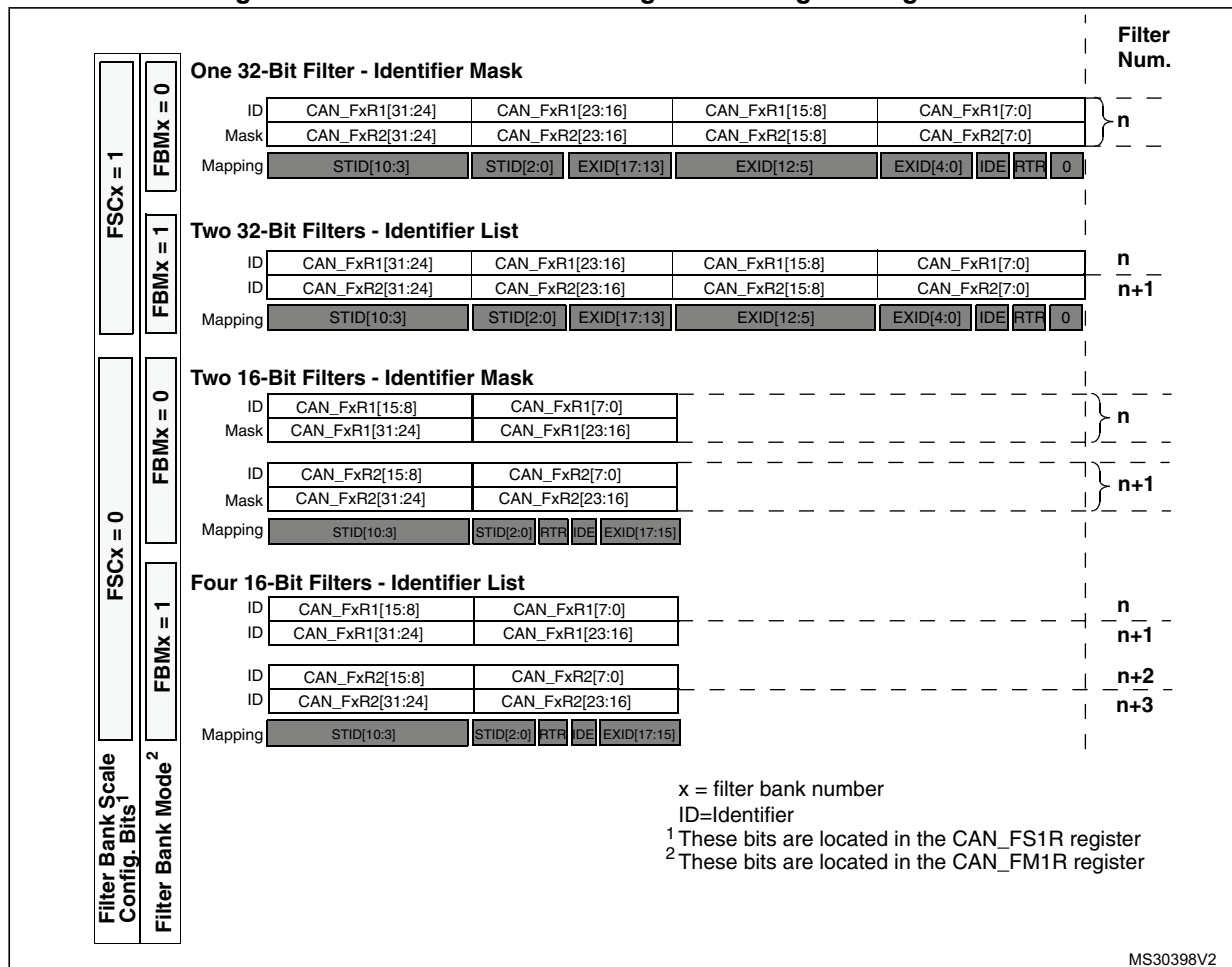
To select single identifiers, configure the Mask/Identifier registers in identifier list mode.

Filters not used by the application should be left deactivated.

Each filter within a filter bank is numbered (called the *Filter Number*) from 0 to a maximum dependent on the mode and the scale of each of the filter banks.

Concerning the filter configuration, refer to [Figure 382](#).

Figure 382. Filter bank scale configuration - register organization



MS30398V2

Filter match index

Once a message has been received in the FIFO it is available to the application. Typically, application data is copied into SRAM locations. To copy the data to the right location the application has to identify the data by means of the identifier. To avoid this, and to ease the access to the SRAM locations, the CAN controller provides a Filter Match Index.

This index is stored in the mailbox together with the message according to the filter priority rules. Thus each received message has its associated filter match index.

The Filter Match index can be used in two ways:

- Compare the Filter Match index with a list of expected values.
- Use the Filter Match Index as an index on an array to access the data destination location.

For non masked filters, the software no longer has to compare the identifier.

If the filter is masked the software reduces the comparison to the masked bits only.

The index value of the filter number does not take into account the activation state of the filter banks. In addition, two independent numbering schemes are used, one for each FIFO. Refer to [Figure 383](#) for an example.

Figure 383. Example of filter numbering

Filter Bank	FIFO0	Filter Num.	Filter Bank	FIFO1	Filter Num.
0	ID List (32-bit)	0 1	2	ID Mask (16-bit)	0 1
1	ID Mask (32-bit)	2	4	ID List (32-bit)	2 3
3	ID List (16-bit)	3 4 5 6	7	Deactivated ID List (16-bit)	4 5
5	Deactivated ID List (32-bit)	7 8	8	ID Mask (16-bit)	6 7
6	ID Mask (16-bit)	9 10	10	Deactivated ID List (16-bit)	8 9 10 11
9	ID List (32-bit)	11 12	11	ID List (32-bit)	12 13
13	ID Mask (32-bit)	13	12	ID Mask (32-bit)	14

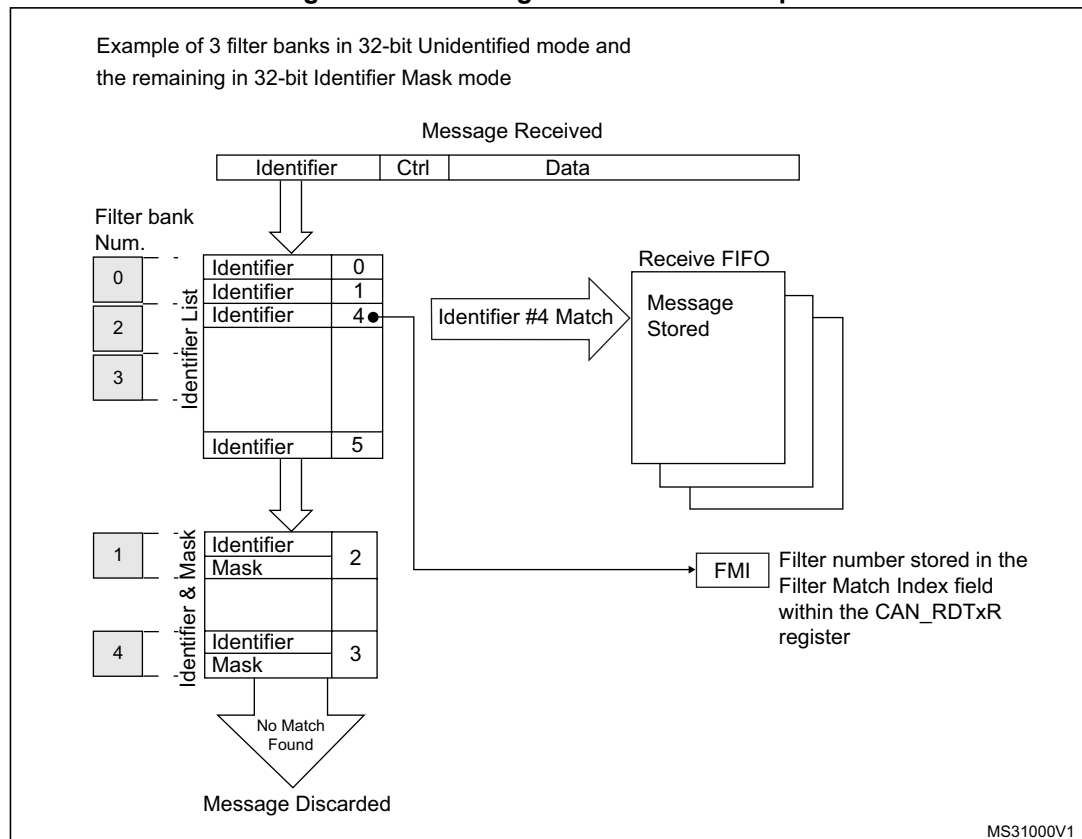
ID=Identifier

MS30399V1

Filter priority rules

Depending on the filter combination it may occur that an identifier passes successfully through several filters. In this case the filter match value stored in the receive mailbox is chosen according to the following priority rules:

- A 32-bit filter takes priority over a 16-bit filter.
- For filters of equal scale, priority is given to the Identifier List mode over the Identifier Mask mode
- For filters of equal scale and mode, priority is given by the filter number (the lower the number, the higher the priority).

Figure 384. Filtering mechanism - example

The example above shows the filtering principle of the bxCAN. On reception of a message, the identifier is compared first with the filters configured in identifier list mode. If there is a match, the message is stored in the associated FIFO and the index of the matching filter is stored in the Filter Match Index. As shown in the example, the identifier matches with Identifier #2 thus the message content and FMI 2 is stored in the FIFO.

If there is no match, the incoming identifier is then compared with the filters configured in mask mode.

If the identifier does not match any of the identifiers configured in the filters, the message is discarded by hardware without disturbing the software.

31.7.5 Message storage

The interface between the software and the hardware for the CAN messages is implemented by means of mailboxes. A mailbox contains all information related to a message; identifier, data, control, status and time stamp information.

Transmit mailbox

The software sets up the message to be transmitted in an empty transmit mailbox. The status of the transmission is indicated by hardware in the CAN_TSR register.

Table 167. Transmit mailbox mapping

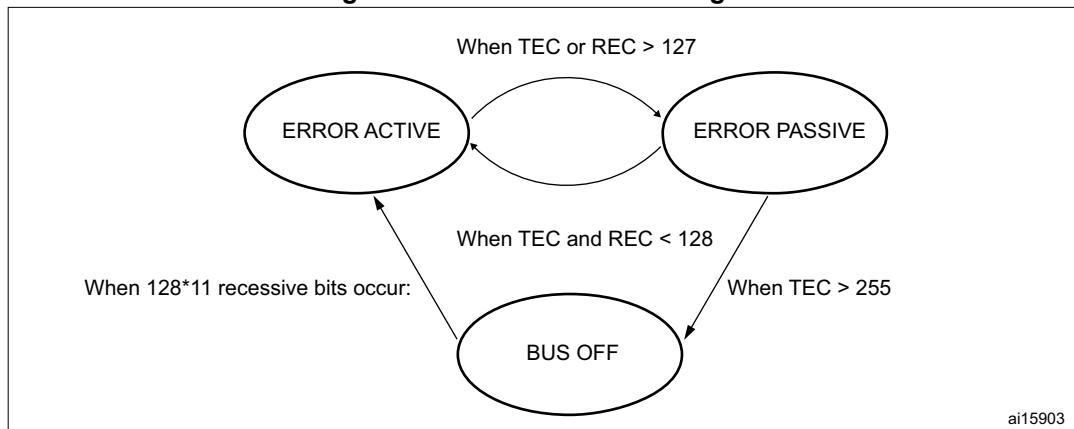
Offset to transmit mailbox base address	Register name
0	CAN_TlRxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

Receive mailbox

When a message has been received, it is available to the software in the FIFO output mailbox. Once the software has handled the message (e.g. read it) the software must release the FIFO output mailbox by means of the RFOM bit in the CAN_RFR register to make the next incoming message available. The filter match index is stored in the MFMI field of the CAN_RDTxR register. The 16-bit time stamp value is stored in the TIME[15:0] field of CAN_RDTxR.

Table 168. Receive mailbox mapping

Offset to receive mailbox base address (bytes)	Register name
0	CAN_RlRxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

Figure 385. CAN error state diagram

31.7.6 Error management

The error management as described in the CAN protocol is handled entirely by hardware using a Transmit Error Counter (TEC value, in CAN_ESR register) and a Receive Error Counter (REC value, in the CAN_ESR register), which get incremented or decremented according to the error condition. For detailed information about TEC and REC management, please refer to the CAN standard.

Both of them may be read by software to determine the stability of the network. Furthermore, the CAN hardware provides detailed information on the current error status in CAN_ESR register. By means of the CAN_IER register (ERRIE bit, etc.), the software can configure the interrupt generation on error detection in a very flexible way.

Bus-Off recovery

The Bus-Off state is reached when TEC is greater than 255, this state is indicated by BOFF bit in CAN_ESR register. In Bus-Off state, the bxCAN is no longer able to transmit and receive messages.

Depending on the ABOM bit in the CAN_MCR register bxCAN will recover from Bus-Off (become error active again) either automatically or on software request. But in both cases the bxCAN has to wait at least for the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on CANRX).

If ABOM is set, the bxCAN will start the recovering sequence automatically after it has entered Bus-Off state.

If ABOM is cleared, the software must initiate the recovering sequence by requesting bxCAN to enter and to leave initialization mode.

Note: In initialization mode, bxCAN does not monitor the CANRX signal, therefore it cannot complete the recovery sequence. **To recover, bxCAN must be in normal mode.**

31.7.7 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

Its operation may be explained simply by splitting nominal bit time into three segments as follows:

- **Synchronization segment (SYNC_SEG):** a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ($1 \times t_{CAN}$).
- **Bit segment 1 (BS1):** defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.
- **Bit segment 2 (BS2):** defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.

The resynchronization Jump Width (SJW) defines an upper bound to the amount of lengthening or shortening of the bit segments. It is programmable between 1 and 4 time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the Bit Timing Register (CAN_BTR) is only possible while the device is in Standby mode.

Note: For a detailed description of the CAN bit timing and resynchronization mechanism, please refer to the ISO 11898 standard.

Figure 386. Bit timing

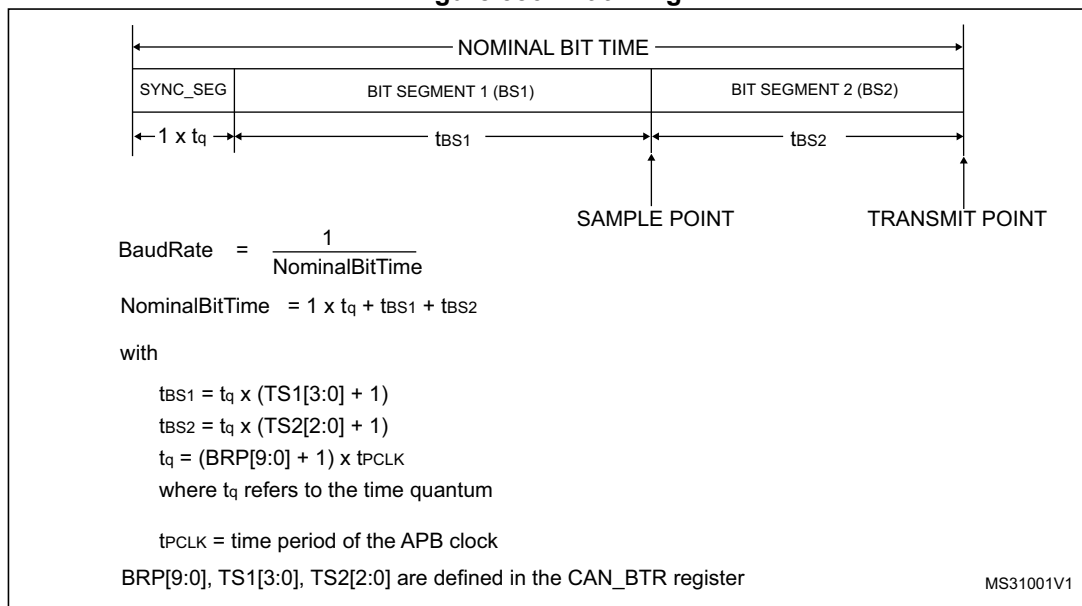
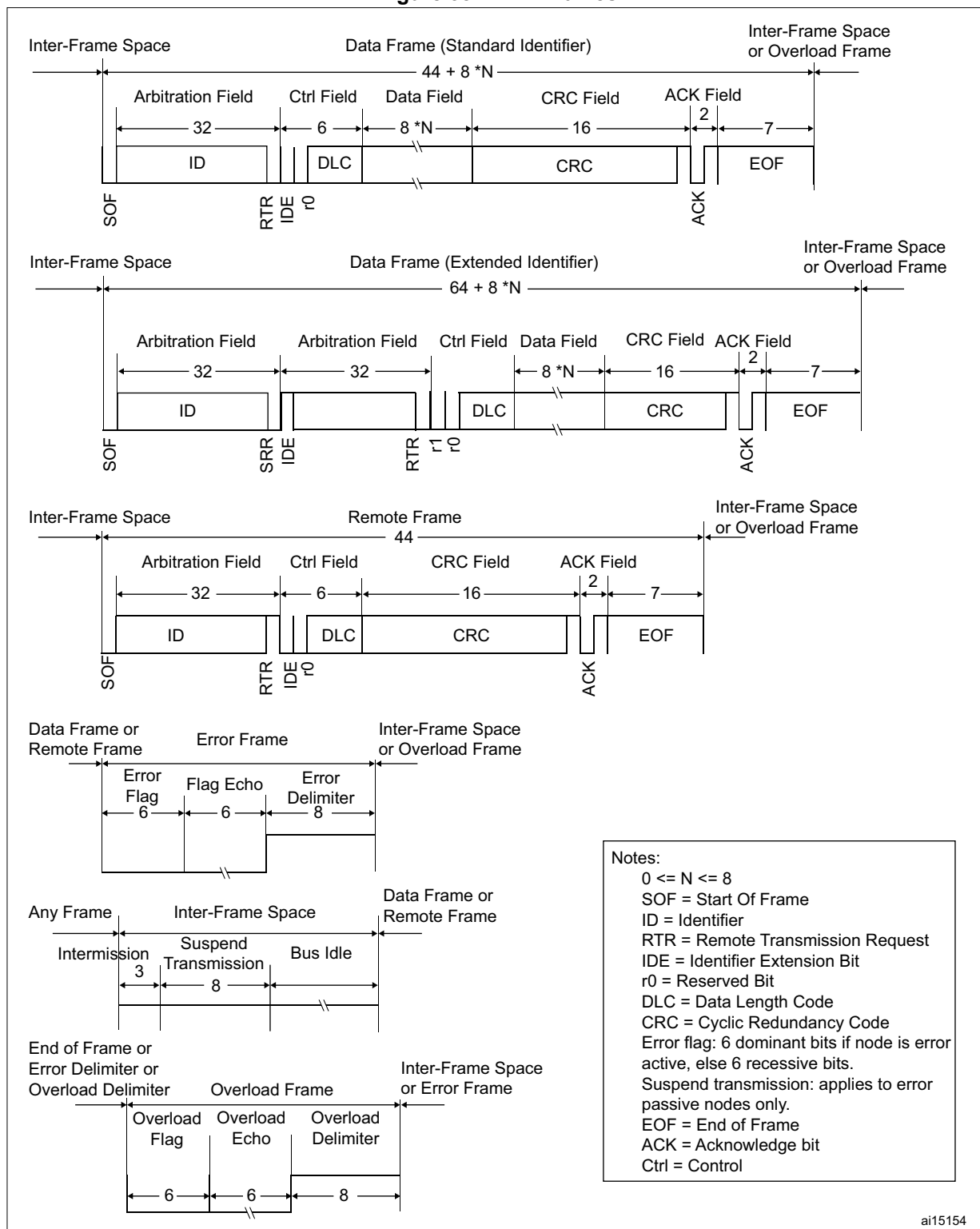


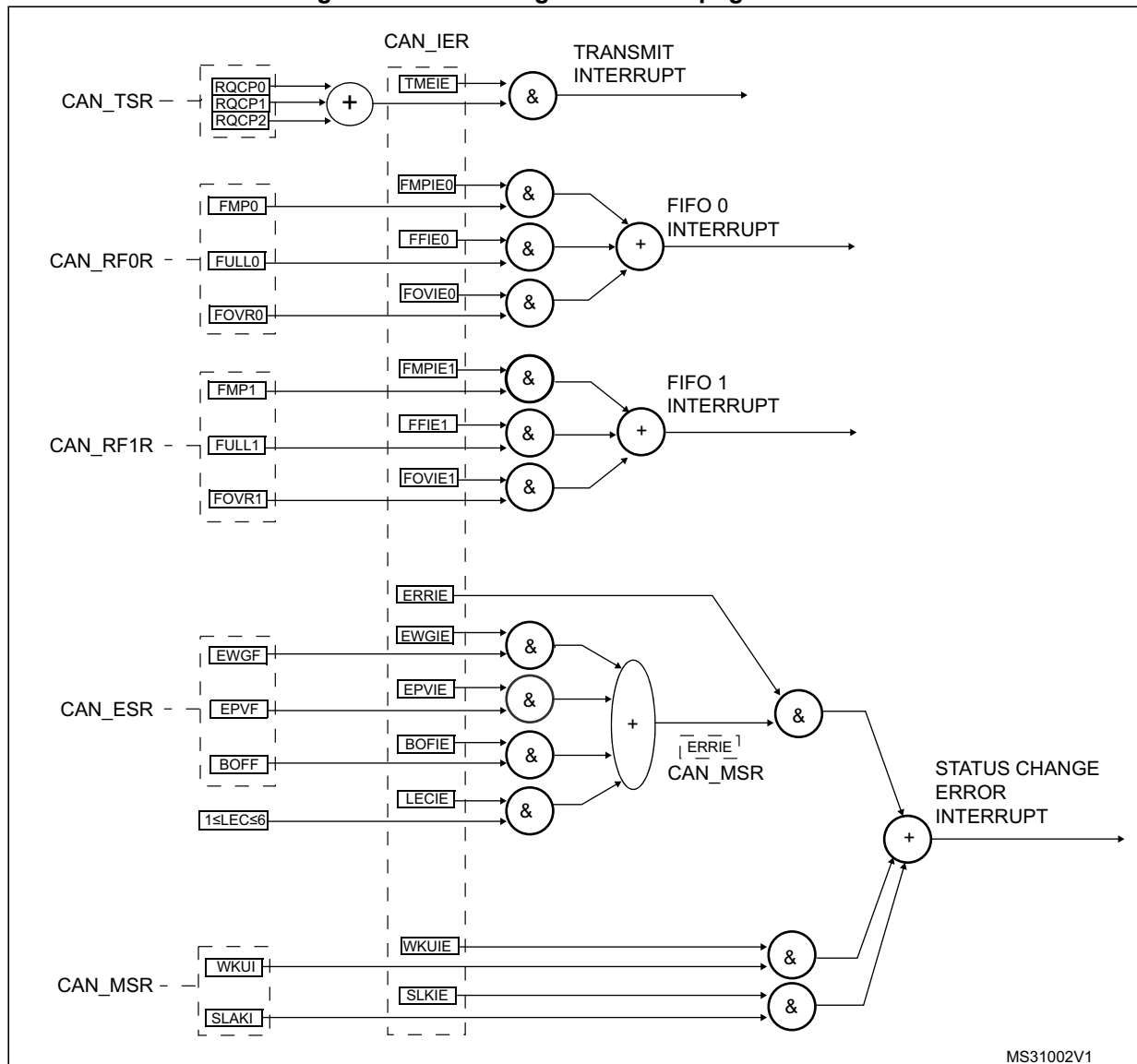
Figure 387. CAN frames



31.8 bxCAN interrupts

Four interrupt vectors are dedicated to bxCAN. Each interrupt source can be independently enabled or disabled by means of the CAN Interrupt Enable Register (CAN_IER).

Figure 388. Event flags and interrupt generation



- The **transmit interrupt** can be generated by the following events:
 - Transmit mailbox 0 becomes empty, RQCP0 bit in the CAN_TSR register set.
 - Transmit mailbox 1 becomes empty, RQCP1 bit in the CAN_TSR register set.
 - Transmit mailbox 2 becomes empty, RQCP2 bit in the CAN_TSR register set.
- The **FIFO 0 interrupt** can be generated by the following events:
 - Reception of a new message, FMP0 bits in the CAN_RF0R register are not '00'.
 - FIFO0 full condition, FULL0 bit in the CAN_RF0R register set.
 - FIFO0 overrun condition, FOVR0 bit in the CAN_RF0R register set.

- The **FIFO 1 interrupt** can be generated by the following events:
 - Reception of a new message, FMP1 bits in the CAN_RF1R register are not '00'.
 - FIFO1 full condition, FULL1 bit in the CAN_RF1R register set.
 - FIFO1 overrun condition, FOVR1 bit in the CAN_RF1R register set.
- The **error and status change interrupt** can be generated by the following events:
 - Error condition, for more details on error conditions please refer to the CAN Error Status register (CAN_ESR).
 - Wakeup condition, SOF monitored on the CAN Rx signal.
 - Entry into Sleep mode.

31.9 CAN registers

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

31.9.1 Register access protection

Erroneous access to certain configuration registers can cause the hardware to temporarily disturb the whole CAN network. Therefore the CAN_BTR register can be modified by software only while the CAN hardware is in initialization mode.

Although the transmission of incorrect data will not cause problems at the CAN network level, it can severely disturb the application. A transmit mailbox can be only modified by software while it is in empty state, refer to [Figure 380: Transmit mailbox states](#).

The filter values can be modified either deactivating the associated filter banks or by setting the FINIT bit. Moreover, the modification of the filter configuration (scale, mode and FIFO assignment) in CAN_FMR, CAN_FSR and CAN_FFR registers can only be done when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

31.9.2 CAN control and status registers

Refer to [Section 2.1](#) for a list of abbreviations used in register descriptions.

CAN master control register (CAN_MCR)

Address offset: 0x00

Reset value: 0x0001 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBF
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	Res	Res	Res	Res	Res	Res	Res	TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 16 **DBF**: Debug freeze

0: CAN working during debug

1: CAN reception/transmission frozen during debug. Reception FIFOs can still be accessed/controlled normally.

Bit 15 **RESET**: bxCAN software master reset

0: Normal operation.

1: Force a master reset of the bxCAN -> Sleep mode activated after reset (FMP bits and CAN_MCR register are initialized to the reset values). This bit is automatically reset to 0.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **TTCM**: Time triggered communication mode

0: Time Triggered Communication mode disabled.

1: Time Triggered Communication mode enabled

Note: For more information on Time Triggered Communication mode, please refer to [Section 31.7.2: Time triggered communication mode](#).

Bit 6 **ABOM**: Automatic bus-off management

This bit controls the behavior of the CAN hardware on leaving the Bus-Off state.

0: The Bus-Off state is left on software request, once 128 occurrences of 11 recessive bits have been monitored and the software has first set and cleared the INRQ bit of the CAN_MCR register.

1: The Bus-Off state is left automatically by hardware once 128 occurrences of 11 recessive bits have been monitored.

For detailed information on the Bus-Off state please refer to [Section 31.7.6: Error management](#).

Bit 5 **AWUM**: Automatic wakeup mode

This bit controls the behavior of the CAN hardware on message reception during Sleep mode.

0: The Sleep mode is left on software request by clearing the SLEEP bit of the CAN_MCR register.

1: The Sleep mode is left automatically by hardware on CAN message detection.

The SLEEP bit of the CAN_MCR register and the SLAK bit of the CAN_MSR register are cleared by hardware.

Bit 4 **NART**: No automatic retransmission

0: The CAN hardware will automatically retransmit the message until it has been successfully transmitted according to the CAN standard.

1: A message will be transmitted only once, independently of the transmission result (successful, error or arbitration lost).

Bit 3 **RFLM**: Receive FIFO locked mode

0: Receive FIFO not locked on overrun. Once a receive FIFO is full the next incoming message will overwrite the previous one.

1: Receive FIFO locked against overrun. Once a receive FIFO is full the next incoming message will be discarded.

Bit 2 TXFP: Transmit FIFO priority

This bit controls the transmission order when several mailboxes are pending at the same time.

0: Priority driven by the identifier of the message

1: Priority driven by the request order (chronologically)

Bit 1 SLEEP: Sleep mode request

This bit is set by software to request the CAN hardware to enter the Sleep mode. Sleep mode will be entered as soon as the current CAN activity (transmission or reception of a CAN frame) has been completed.

This bit is cleared by software to exit Sleep mode.

This bit is cleared by hardware when the AWUM bit is set and a SOF bit is detected on the CAN Rx signal.

This bit is set after reset - CAN starts in Sleep mode.

Bit 0 INRQ: Initialization request

The software clears this bit to switch the hardware into normal mode. Once 11 consecutive recessive bits have been monitored on the Rx signal the CAN hardware is synchronized and ready for transmission and reception. Hardware signals this event by clearing the INAK bit in the CAN_MSR register.

Software sets this bit to request the CAN hardware to enter initialization mode. Once software has set the INRQ bit, the CAN hardware waits until the current CAN activity (transmission or reception) is completed before entering the initialization mode. Hardware signals this event by setting the INAK bit in the CAN_MSR register.

CAN master status register (CAN_MSR)

Address offset: 0x04

Reset value: 0x0000 0C02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	RX	SAMP	RXM	TXM	Res	Res	Res	SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

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

Bit 11 RX: CAN Rx signal

Monitors the actual value of the **CAN_RX** Pin.

Bit 10 SAMP: Last sample point

The value of RX on the last sample point (current received bit value).

Bit 9 RXM: Receive mode

The CAN hardware is currently receiver.

Bit 8 TXM: Transmit mode

The CAN hardware is currently transmitter.

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

Bit 4 SLAKI: Sleep acknowledge interrupt

When SLKIE=1, this bit is set by hardware to signal that the bxCAN has entered Sleep Mode. When set, this bit generates a status change interrupt if the SLKIE bit in the CAN_IER register is set.

This bit is cleared by software or by hardware, when SLAK is cleared.

Note: When SLKIE=0, no polling on SLAKI is possible. In this case the SLAK bit can be polled.

Bit 3 WKUI: Wakeup interrupt

This bit is set by hardware to signal that a SOF bit has been detected while the CAN hardware was in Sleep mode. Setting this bit generates a status change interrupt if the WKUIE bit in the CAN_IER register is set.

This bit is cleared by software.

Bit 2 ERRI: Error interrupt

This bit is set by hardware when a bit of the CAN_ESR has been set on error detection and the corresponding interrupt in the CAN_IER is enabled. Setting this bit generates a status change interrupt if the ERRIE bit in the CAN_IER register is set.

This bit is cleared by software.

Bit 1 SLAK: Sleep acknowledge

This bit is set by hardware and indicates to the software that the CAN hardware is now in Sleep mode. This bit acknowledges the Sleep mode request from the software (set SLEEP bit in CAN_MCR register).

This bit is cleared by hardware when the CAN hardware has left Sleep mode (to be synchronized on the CAN bus). To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.

Note: The process of leaving Sleep mode is triggered when the SLEEP bit in the CAN_MCR register is cleared. Please refer to the AWUM bit of the CAN_MCR register description for detailed information for clearing SLEEP bit

Bit 0 INAK: Initialization acknowledge

This bit is set by hardware and indicates to the software that the CAN hardware is now in initialization mode. This bit acknowledges the initialization request from the software (set INRQ bit in CAN_MCR register).

This bit is cleared by hardware when the CAN hardware has left the initialization mode (to be synchronized on the CAN bus). To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.

CAN transmit status register (CAN_TSR)

Address offset: 0x08

Reset value: 0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE[1:0]		ABRQ2	Res	Res	Res	TERR2	ALST2	TXOK2	RQCP2
r	r	r	r	r	r	r	r	rs				rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRQ1	Res	Res	Res	TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Res	Res	Res	TERR0	ALST0	TXOK0	RQCP0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

- Bit 31 **LOW2**: Lowest priority flag for mailbox 2
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 2 has the lowest priority.
- Bit 30 **LOW1**: Lowest priority flag for mailbox 1
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 1 has the lowest priority.
- Bit 29 **LOW0**: Lowest priority flag for mailbox 0
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 0 has the lowest priority.
Note: The LOW[2:0] bits are set to zero when only one mailbox is pending.
- Bit 28 **TME2**: Transmit mailbox 2 empty
This bit is set by hardware when no transmit request is pending for mailbox 2.
- Bit 27 **TME1**: Transmit mailbox 1 empty
This bit is set by hardware when no transmit request is pending for mailbox 1.
- Bit 26 **TME0**: Transmit mailbox 0 empty
This bit is set by hardware when no transmit request is pending for mailbox 0.
- Bits 25:24 **CODE[1:0]**: Mailbox code
In case at least one transmit mailbox is free, the code value is equal to the number of the next transmit mailbox free.
In case all transmit mailboxes are pending, the code value is equal to the number of the transmit mailbox with the lowest priority.
- Bit 23 **ABRQ2**: Abort request for mailbox 2
Set by software to abort the transmission request for the corresponding mailbox.
Cleared by hardware when the mailbox becomes empty.
Setting this bit has no effect when the mailbox is not pending for transmission.
- Bits 22:20 Reserved, must be kept at reset value.
- Bit 19 **TERR2**: Transmission error of mailbox 2
This bit is set when the previous TX failed due to an error.
- Bit 18 **ALST2**: Arbitration lost for mailbox 2
This bit is set when the previous TX failed due to an arbitration lost.
- Bit 17 **TXOK2**: Transmission OK of mailbox 2
The hardware updates this bit after each transmission attempt.
0: The previous transmission failed
1: The previous transmission was successful
This bit is set by hardware when the transmission request on mailbox 2 has been completed successfully. Please refer to [Figure 380](#).
- Bit 16 **RQCP2**: Request completed mailbox2
Set by hardware when the last request (transmit or abort) has been performed.
Cleared by software writing a "1" or by hardware on transmission request (TXRQ2 set in CAN_TMD2R register).
Clearing this bit clears all the status bits (TXOK2, ALST2 and TERR2) for Mailbox 2.
- Bit 15 **ABRQ1**: Abort request for mailbox 1
Set by software to abort the transmission request for the corresponding mailbox.
Cleared by hardware when the mailbox becomes empty.
Setting this bit has no effect when the mailbox is not pending for transmission.
- Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **TERR1**: Transmission error of mailbox1

This bit is set when the previous TX failed due to an error.

Bit 10 **ALST1**: Arbitration lost for mailbox1

This bit is set when the previous TX failed due to an arbitration lost.

Bit 9 **TXOK1**: Transmission OK of mailbox1

The hardware updates this bit after each transmission attempt.

0: The previous transmission failed

1: The previous transmission was successful

This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Please refer to [Figure 380](#)

Bit 8 **RQCP1**: Request completed mailbox1

Set by hardware when the last request (transmit or abort) has been performed.

Cleared by software writing a "1" or by hardware on transmission request (TXRQ1 set in CAN_TI1R register).

Clearing this bit clears all the status bits (TXOK1, ALST1 and TERR1) for Mailbox 1.

Bit 7 **ABRQ0**: Abort request for mailbox0

Set by software to abort the transmission request for the corresponding mailbox.

Cleared by hardware when the mailbox becomes empty.

Setting this bit has no effect when the mailbox is not pending for transmission.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **TERR0**: Transmission error of mailbox0

This bit is set when the previous TX failed due to an error.

Bit 2 **ALST0**: Arbitration lost for mailbox0

This bit is set when the previous TX failed due to an arbitration lost.

Bit 1 **TXOK0**: Transmission OK of mailbox0

The hardware updates this bit after each transmission attempt.

0: The previous transmission failed

1: The previous transmission was successful

This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Please refer to [Figure 380](#)

Bit 0 **RQCP0**: Request completed mailbox0

Set by hardware when the last request (transmit or abort) has been performed.

Cleared by software writing a "1" or by hardware on transmission request (TXRQ0 set in CAN_TI0R register).

Clearing this bit clears all the status bits (TXOK0, ALST0 and TERR0) for Mailbox 0.

CAN receive FIFO 0 register (CAN_RF0R)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	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	RFOM0	FOVR0	FULL0	Res	FMP0[1:0]	
										rs	rc_w1	rc_w1		r	r

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

Bit 5 **RFOM0**: Release FIFO 0 output mailbox

Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If at least two messages are pending in the FIFO, the software has to release the output mailbox to access the next message.

Cleared by hardware when the output mailbox has been released.

Bit 4 **FOVR0**: FIFO 0 overrun

This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.

This bit is cleared by software.

Bit 3 **FULL0**: FIFO 0 full

Set by hardware when three messages are stored in the FIFO.

This bit is cleared by software.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FMP0[1:0]**: FIFO 0 message pending

These bits indicate how many messages are pending in the receive FIFO.

FMP is increased each time the hardware stores a new message in to the FIFO. FMP is decreased each time the software releases the output mailbox by setting the RFOM0 bit.

CAN receive FIFO 1 register (CAN_RF1R)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	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	RFOM1	FOVR1	FULL1	Res	FMP1[1:0]	
										rs	rc_w1	rc_w1		r	r

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

Bit 5 **RFOM1**: Release FIFO 1 output mailbox

Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If at least two messages are pending in the FIFO, the software has to release the output mailbox to access the next message.

Cleared by hardware when the output mailbox has been released.

Bit 4 **FOVR1**: FIFO 1 overrun

This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.

This bit is cleared by software.

Bit 3 **FULL1**: FIFO 1 full

Set by hardware when three messages are stored in the FIFO.
This bit is cleared by software.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FMP1[1:0]**: FIFO 1 message pending

These bits indicate how many messages are pending in the receive FIFO1.
FMP1 is increased each time the hardware stores a new message in to the FIFO1. FMP is decreased each time the software releases the output mailbox by setting the RFOM1 bit.

CAN interrupt enable register (CAN_IER)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SLKIE	WKUIE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Res	Res	Res	LEC IE	BOF IE	EPV IE	EWG IE	Res	FOV IE1	FF IE1	FMP IE1	FOV IE0	FF IE0	FMP IE0	TME IE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **SLKIE**: Sleep interrupt enable

0: No interrupt when SLAKI bit is set.
1: Interrupt generated when SLAKI bit is set.

Bit 16 **WKUIE**: Wakeup interrupt enable

0: No interrupt when WKUI is set.
1: Interrupt generated when WKUI bit is set.

Bit 15 **ERRIE**: Error interrupt enable

0: No interrupt will be generated when an error condition is pending in the CAN_ESR.
1: An interrupt will be generation when an error condition is pending in the CAN_ESR.

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

Bit 11 **LECIE**: Last error code interrupt enable

0: ERRI bit will not be set when the error code in LEC[2:0] is set by hardware on error detection.
1: ERRI bit will be set when the error code in LEC[2:0] is set by hardware on error detection.

Bit 10 **BOFIE**: Bus-off interrupt enable

0: ERRI bit will not be set when BOFF is set.
1: ERRI bit will be set when BOFF is set.

Bit 9 **EPVIE**: Error passive interrupt enable

0: ERRI bit will not be set when EPVF is set.
1: ERRI bit will be set when EPVF is set.

- Bit 8 **EWGIE**: Error warning interrupt enable
 0: ERRI bit will not be set when EWGF is set.
 1: ERRI bit will be set when EWGF is set.
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **FOVIE1**: FIFO overrun interrupt enable
 0: No interrupt when FOVR is set.
 1: Interrupt generation when FOVR is set.
- Bit 5 **FFIE1**: FIFO full interrupt enable
 0: No interrupt when FULL bit is set.
 1: Interrupt generated when FULL bit is set.
- Bit 4 **FMPIE1**: FIFO message pending interrupt enable
 0: No interrupt generated when state of FMP[1:0] bits are not 00b.
 1: Interrupt generated when state of FMP[1:0] bits are not 00b.
- Bit 3 **FOVIE0**: FIFO overrun interrupt enable
 0: No interrupt when FOVR bit is set.
 1: Interrupt generated when FOVR bit is set.
- Bit 2 **FFIE0**: FIFO full interrupt enable
 0: No interrupt when FULL bit is set.
 1: Interrupt generated when FULL bit is set.
- Bit 1 **FMPIE0**: FIFO message pending interrupt enable
 0: No interrupt generated when state of FMP[1:0] bits are not 00b.
 1: Interrupt generated when state of FMP[1:0] bits are not 00b.
- Bit 0 **TMEIE**: Transmit mailbox empty interrupt enable
 0: No interrupt when RQCPx bit is set.
 1: Interrupt generated when RQCPx bit is set.
- Note: Refer to [Section 31.8: bxCAN interrupts](#).*

CAN error status register (CAN_ESR)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bits 31:24 **REC[7:0]**: Receive error counter

The implementing part of the fault confinement mechanism of the CAN protocol. In case of an error during reception, this counter is incremented by 1 or by 8 depending on the error condition as defined by the CAN standard. After every successful reception the counter is decremented by 1 or reset to 120 if its value was higher than 128. When the counter value exceeds 127, the CAN controller enters the error passive state.

Bits 23:16 **TEC[7:0]**: Least significant byte of the 9-bit transmit error counter

The implementing part of the fault confinement mechanism of the CAN protocol.

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

Bits 6:4 **LEC[2:0]**: Last error code

This field is set by hardware and holds a code which indicates the error condition of the last error detected on the CAN bus. If a message has been transferred (reception or transmission) without error, this field will be cleared to '0'.

The LEC[2:0] bits can be set to value 0b111 by software. They are updated by hardware to indicate the current communication status.

000: No Error

001: Stuff Error

010: Form Error

011: Acknowledgment Error

100: Bit recessive Error

101: Bit dominant Error

110: CRC Error

111: Set by software

Bit 3 Reserved, must be kept at reset value.

Bit 2 **BOFF**: Bus-off flag

This bit is set by hardware when it enters the bus-off state. The bus-off state is entered on TEC overflow, greater than 255, refer to [Section 31.7.6 on page 975](#).

Bit 1 **EPVF**: Error passive flag

This bit is set by hardware when the Error Passive limit has been reached (Receive Error Counter or Transmit Error Counter > 127).

Bit 0 **EWGF**: Error warning flag

This bit is set by hardware when the warning limit has been reached (Receive Error Counter or Transmit Error Counter ≥ 96).

CAN bit timing register (CAN_BTR)

Address offset: 0x1C

Reset value: 0x0123 0000

This register can only be accessed by the software when the CAN hardware is in initialization mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SILM	LBKM	Res	Res	Res	Res	SJW[1:0]		Res	TS2[2:0]			TS1[3:0]			
rw	rw					rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	BRP[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- Bit 31 **SILM**: Silent mode (debug)
 0: Normal operation
 1: Silent Mode
- Bit 30 **LBKM**: Loop back mode (debug)
 0: Loop Back Mode disabled
 1: Loop Back Mode enabled
- Bits 29:26 Reserved, must be kept at reset value.
- Bits 25:24 **SJW[1:0]**: Resynchronization jump width
 These bits define the maximum number of time quanta the CAN hardware is allowed to lengthen or shorten a bit to perform the resynchronization.
 $t_{RJW} = t_{CAN} \times (SJW[1:0] + 1)$
- Bit 23 Reserved, must be kept at reset value.
- Bits 22:20 **TS2[2:0]**: Time segment 2
 These bits define the number of time quanta in Time Segment 2.
 $t_{BS2} = t_{CAN} \times (TS2[2:0] + 1)$
- Bits 19:16 **TS1[3:0]**: Time segment 1
 These bits define the number of time quanta in Time Segment 1
 $t_{BS1} = t_{CAN} \times (TS1[3:0] + 1)$
 For more information on bit timing, please refer to [Section 31.7.7: Bit timing on page 975](#).
- Bits 15:10 Reserved, must be kept at reset value.
- Bits 9:0 **BRP[9:0]**: Baud rate prescaler
 These bits define the length of a time quanta.
 $t_q = (BRP[9:0] + 1) \times t_{PCLK}$

31.9.3 CAN mailbox registers

This chapter describes the registers of the transmit and receive mailboxes. Refer to [Section 31.7.5: Message storage on page 973](#) for detailed register mapping.

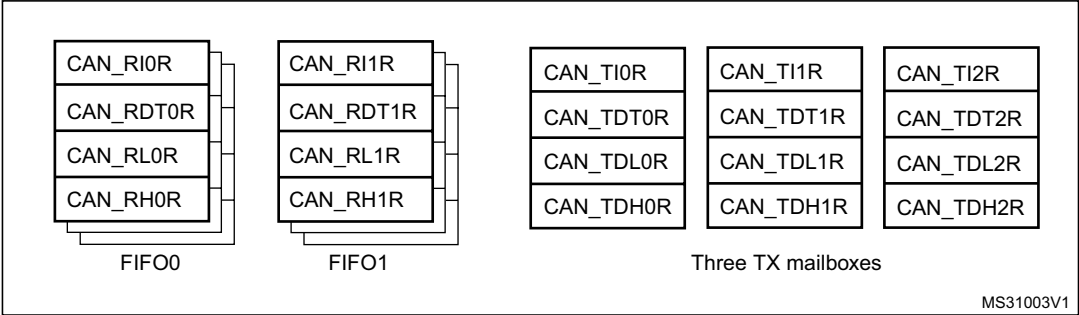
Transmit and receive mailboxes have the same registers except:

- The FMI field in the CAN_RDTxR register.
- A receive mailbox is always write protected.
- A transmit mailbox is write-enabled only while empty, corresponding TME bit in the CAN_TSR register set.

There are 3 TX Mailboxes and 2 RX Mailboxes. Each RX Mailbox allows access to a 3 level depth FIFO, the access being offered only to the oldest received message in the FIFO.

Each mailbox consist of 4 registers.

Figure 389. Can mailbox registers



CAN TX mailbox identifier register (CAN_TlRxR) (x = 0..2)

Address offsets: 0x180, 0x190, 0x1A0

Reset value: 0xFFFF XXXX (except bit 0, TXRQ = 0)

All TX registers are write protected when the mailbox is pending transmission (TMEx reset).

This register also implements the TX request control (bit 0) - reset value 0.

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

Bits 31:21 **STID[10:0]/EXID[28:18]**: Standard identifier or extended identifier

The standard identifier or the MSBs of the extended identifier (depending on the IDE bit value).

Bit 20:3 **EXID[17:0]**: Extended identifier

The LSBs of the extended identifier.

Bit 2 **IDE**: Identifier extension

This bit defines the identifier type of message in the mailbox.

0: Standard identifier.

1: Extended identifier.

Bit 1 **RTR**: Remote transmission request

0: Data frame

1: Remote frame

Bit 0 **TXRQ**: Transmit mailbox request

Set by software to request the transmission for the corresponding mailbox.

Cleared by hardware when the mailbox becomes empty.

CAN mailbox data length control and time stamp register (CAN_TDTxR) (x = 0..2)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x184, 0x194, 0x1A4

Reset value: 0xFFFF XXXX

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

Bits 31:16 **TIME[15:0]**: Message time stamp

This field contains the 16-bit timer value captured at the SOF transmission.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TGT**: Transmit global time

This bit is active only when the hardware is in the Time Trigger Communication mode, TTCM bit of the CAN_MCR register is set.

0: Time stamp TIME[15:0] is not sent.

1: Time stamp TIME[15:0] value is sent in the last two data bytes of the 8-byte message: TIME[7:0] in data byte 7 and TIME[15:8] in data byte 6, replacing the data written in CAN_TDHxR[31:16] register (DATA6[7:0] and DATA7[7:0]). DLC must be programmed as 8 in order these two bytes to be sent over the CAN bus.

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

Bits 3:0 **DLC[3:0]**: Data length code

This field defines the number of data bytes a data frame contains or a remote frame request. A message can contain from 0 to 8 data bytes, depending on the value in the DLC field.

CAN mailbox data low register (CAN_TDLxR) (x = 0..2)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x188, 0x198, 0x1A8

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DATA3[7:0]**: Data byte 3

Data byte 3 of the message.

Bits 23:16 **DATA2[7:0]**: Data byte 2

Data byte 2 of the message.

Bits 15:8 **DATA1[7:0]**: Data byte 1

Data byte 1 of the message.

Bits 7:0 **DATA0[7:0]**: Data byte 0

Data byte 0 of the message.

A message can contain from 0 to 8 data bytes and starts with byte 0.

CAN mailbox data high register (CAN_TDHxR) (x = 0..2)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x18C, 0x19C, 0x1AC

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DATA7[7:0]**: Data byte 7

Data byte 7 of the message.

Note: If TGT of this message and TTCM are active, DATA7 and DATA6 will be replaced by the TIME stamp value.

Bits 23:16 **DATA6[7:0]**: Data byte 6

Data byte 6 of the message.

Bits 15:8 **DATA5[7:0]**: Data byte 5

Data byte 5 of the message.

Bits 7:0 **DATA4[7:0]**: Data byte 4

Data byte 4 of the message.

CAN receive FIFO mailbox identifier register (CAN_RlRxR) (x = 0..1)

Address offsets: 0x1B0, 0x1C0

Reset value: 0xFFFF XXXX

All RX registers are write protected.

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

Bits 31:21 **STID[10:0]/EXID[28:18]**: Standard identifier or extended identifier

The standard identifier or the MSBs of the extended identifier (depending on the IDE bit value).

Bits 20:3 **EXID[17:0]**: Extended identifier

The LSBs of the extended identifier.

Bit 2 **IDE**: Identifier extension

This bit defines the identifier type of message in the mailbox.

0: Standard identifier.

1: Extended identifier.

Bit 1 **RTR**: Remote transmission request

0: Data frame

1: Remote frame

Bit 0 Reserved, must be kept at reset value.

CAN receive FIFO mailbox data length control and time stamp register (CAN_RDTxR) (x = 0..1)

Address offsets: 0x1B4, 0x1C4

Reset value: 0xFFFF XXXX

All RX registers are write protected.

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

Bits 31:16 **TIME[15:0]**: Message time stamp

This field contains the 16-bit timer value captured at the SOF detection.

Bits 15:8 **FMI[7:0]**: Filter match index

This register contains the index of the filter the message stored in the mailbox passed through. For more details on identifier filtering please refer to [Section 31.7.4: Identifier filtering on page 969](#) - **Filter Match Index** paragraph.

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

Bits 3:0 **DLC[3:0]**: Data length code

This field defines the number of data bytes a data frame contains (0 to 8). It is 0 in the case of a remote frame request.

CAN receive FIFO mailbox data low register (CAN_RDLxR) (x = 0..1)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x1B8, 0x1C8

Reset value: 0xFFFF XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DATA3[7:0]**: Data Byte 3

Data byte 3 of the message.

Bits 23:16 **DATA2[7:0]**: Data Byte 2

Data byte 2 of the message.

Bits 15:8 **DATA1[7:0]**: Data Byte 1

Data byte 1 of the message.

Bits 7:0 **DATA0[7:0]**: Data Byte 0

Data byte 0 of the message.

A message can contain from 0 to 8 data bytes and starts with byte 0.

CAN receive FIFO mailbox data high register (CAN_RDHxR) (x = 0..1)

Address offsets: 0x1BC, 0x1CC

Reset value: 0xFFFF XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DATA7[7:0]**: Data Byte 7

Data byte 3 of the message.

Bits 23:16 **DATA6[7:0]**: Data Byte 6
Data byte 2 of the message.

Bits 15:8 **DATA5[7:0]**: Data Byte 5
Data byte 1 of the message.

Bits 7:0 **DATA4[7:0]**: Data Byte 4
Data byte 0 of the message.

31.9.4 CAN filter registers

CAN filter master register (CAN_FMR)

Address offset: 0x200
Reset value: 0x2A1C 0E01

All bits of this register are set and cleared by software.

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

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

Bit 0 **FINIT**: Filter initialization mode
Initialization mode for filter banks
0: Active filters mode.
1: Initialization mode for the filters.



CAN filter mode register (CAN_FM1R)

Address offset: 0x204

Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	FBM13	FBM12	FBM11	FBM10	FBM9	FBM8	FBM7	FBM6	FBM5	FBM4	FBM3	FBM2	FBM1	FBM0
		rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Note: Please refer to [Figure 382: Filter bank scale configuration - register organization on page 971](#)

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

Bits 13:0 **FBMx**: Filter mode

Mode of the registers of Filter x.

0: Two 32-bit registers of filter bank x are in Identifier Mask mode.

1: Two 32-bit registers of filter bank x are in Identifier List mode.

Note: Bits 27:14 are available in connectivity line devices only and are reserved otherwise.

CAN filter scale register (CAN_FS1R)

Address offset: 0x20C

Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	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	FSC13	FSC12	FSC11	FSC10	FSC9	FSC8	FSC7	FSC6	FSC5	FSC4	FSC3	FSC2	FSC1	FSC0
		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 **FSCx**: Filter scale configuration

These bits define the scale configuration of Filters 13-0.

0: Dual 16-bit scale configuration

1: Single 32-bit scale configuration

Note: Bits 27:14 are available in connectivity line devices only and are reserved otherwise.

Note: Please refer to [Figure 382: Filter bank scale configuration - register organization on page 971](#).

CAN filter FIFO assignment register (CAN_FFA1R)

Address offset: 0x214

Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	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	FFA13	FFA12	FFA11	FFA10	FFA9	FFA8	FFA7	FFA6	FFA5	FFA4	FFA3	FFA2	FFA1	FFA0
		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 **FFAx**: Filter FIFO assignment for filter x

The message passing through this filter will be stored in the specified FIFO.

0: Filter assigned to FIFO 0

1: Filter assigned to FIFO 1

Note: Bits 27:14 are available in connectivity line devices only and are reserved otherwise.

CAN filter activation register (CAN_FA1R)

Address offset: 0x21C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	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	FACT1 3	FACT1 2	FACT1 1	FACT1 0	FACT9	FACT8	FACT7	FACT6	FACT5	FACT4	FACT3	FACT2	FACT1	FACT0
		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 **FACTx**: Filter active

The software sets this bit to activate Filter x. To modify the Filter x registers (CAN_FxR[0:7]), the FACTx bit must be cleared or the FINIT bit of the CAN_FMR register must be set.

0: Filter x is not active

1: Filter x is active

Note: Bits 27:14 are available in connectivity line devices only and are reserved otherwise.

Filter bank i register x (CAN_FiRx) (i = 0..13, x = 1, 2)

Address offsets: 0x240 to 0x2AC

Reset value: 0xFFFF XXXX

There are 14 filter banks, $i = 0$ to 13. Each filter bank i is composed of two 32-bit registers, CAN_FiR[2:1].

This register can only be modified when the FACTx bit of the CAN_FAxR register is cleared or when the FINIT bit of the CAN_FMR register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
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
FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

In all configurations:

Bits 31:0 **FB[31:0]**: Filter bits

Identifier

Each bit of the register specifies the level of the corresponding bit of the expected identifier.

0: Dominant bit is expected

1: Recessive bit is expected

Mask

Each bit of the register specifies whether the bit of the associated identifier register must match with the corresponding bit of the expected identifier or not.

0: Don't care, the bit is not used for the comparison

1: Must match, the bit of the incoming identifier must have the same level as specified in the corresponding identifier register of the filter.

Note: Depending on the scale and mode configuration of the filter the function of each register can differ. For the filter mapping, functions description and mask registers association, refer to [Section 31.7.4: Identifier filtering on page 969](#).

A Mask/Identifier register in **mask mode** has the same bit mapping as in **identifier list mode**.

For the register mapping/addresses of the filter banks please refer to the [Table 169 on page 1001](#).

31.9.5 bxCAN register map

Refer to [Section 3.2.2 on page 48](#) for the register boundary addresses.

Table 169. bxCAN 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	CAN_MCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBF	RESET	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res													
	Reset value																1	0								0	0	0	0	0	0	0	1	0												
0x004	CAN_MSR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RX	SAMP	RXM	TXM	Res	Res	Res	Res	Res	Res	Res	Res													
	Reset value																					1	1	0	0	Res	Res	Res	Res	Res	Res	Res	Res													
0x008	CAN_TSR	LOW[2:0]			TME[2:0]			CODE[1:0]			ABRQ2			TERR2			ALST2			TXOK2			RQCP2			ABRQ1			TERR1			ALST1			TXOK1			RQCP1			ABRQ0			Res		
	Reset value	0	0	0	1	1	1	0	0	0					0	0	0	0	0				0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0				
0x00C	CAN_RF0R	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RFOM0	FOVR0	FULL0	Res	FMP0[1:0]													
	Reset value																											0	0	0			0	0												
0x010	CAN_RF1R	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RFOM1	FOVR1	FULL1	Res	FMP1[1:0]													
	Reset value																											0	0	0			0	0												
0x014	CAN_IER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SLKIE	WKUIE	ERRIE	Res	Res	Res	Res	LECIE	BOFIE	EPVIE	EWGIE	Res	Res	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE											
	Reset value															0	0	0					0	0	0	0		0	0	0	0	0	0	0	0											
0x018	CAN_ESR	REC[7:0]								TEC[7:0]								Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LEC[2:0]			Res	BOFF	EPVF	EWGF										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0	0		0	0	0	0											
0x01C	CAN_BTR	SILM	LBKM	Res	Res	Res	Res	SJW[1:0]		Res	TS2[2:0]			TS1[3:0]			Res	Res	Res	Res	Res	Res	Res	Res	BRP[9:0]																					
	Reset value	0	0					0	0		0	1	0	0	0	1	1								0	0	0	0	0	0	0	0	0	0	0	0										
0x020-0x17F		Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res												
0x180	CAN_TI0R	STID[10:0]/EXID[28:18]												EXID[17:0]																		IDE	RTR	TXRQ												
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0												

Table 169. bxCAN register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x184	CAN_TDT0R	TIME[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x188	CAN_TDL0R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x18C	CAN_TDH0R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x190	CAN_TI1R	STID[10:0]/EXID[28:18]												EXID[17:0]												IDE			RTR	TXRQ			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	
0x194	CAN_TDT1R	TIME[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x198	CAN_TDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x19C	CAN_TDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1A0	CAN_TI2R	STID[10:0]/EXID[28:18]												EXID[17:0]												IDE			RTR	TXRQ			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	
0x1A4	CAN_TDT2R	TIME[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x1A8	CAN_TDL2R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1AC	CAN_TDH2R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1B0	CAN_RI0R	STID[10:0]/EXID[28:18]												EXID[17:0]												IDE			RTR	Res.			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Res.

Table 169. bxCAN register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x1B4	CAN_RDT0R	TIME[15:0]															FMI[7:0]							Res.	Res.	Res.	Res.	DLC[3:0]						
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	
0x1B8	CAN_RDL0R	DATA3[7:0]							DATA2[7:0]							DATA1[7:0]							DATA0[7:0]											
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x1BC	CAN_RDH0R	DATA7[7:0]							DATA6[7:0]							DATA5[7:0]							DATA4[7:0]											
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x1C0	CAN_RI1R	STID[10:0]/EXID[28:18]											EXID[17:0]																	IDE	RTR	Res.		
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
0x1C4	CAN_RDT1R	TIME[15:0]															FMI[7:0]							Res.	Res.	Res.	Res.	DLC[3:0]						
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	
0x1C8	CAN_RDL1R	DATA3[7:0]							DATA2[7:0]							DATA1[7:0]							DATA0[7:0]											
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x1CC	CAN_RDH1R	DATA7[7:0]							DATA6[7:0]							DATA5[7:0]							DATA4[7:0]											
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x1D0-0x1FF		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
0x200	CAN_FMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x204	CAN_FM1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FBM[13:0]														
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x208		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x20C	CAN_FS1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSC[13:0]														
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x210		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x214	CAN_FFA1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FFA[13:0]														
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 169. bxCAN register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x218		Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
0x21C	CAN_FA1R	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FACT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x220		Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
0x224-0x23F		Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
0x240	CAN_F0R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x244	CAN_F0R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x248	CAN_F1R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x24C	CAN_F1R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
.																															
0x318	CAN_F27R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x31C	CAN_F27R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

32 Universal serial bus full-speed device interface (USB)

32.1 Introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB1 bus.

USB suspend/resume are supported which allows to stop the device clocks for low-power consumption.

32.2 USB main features

- USB specification version 2.0 full-speed compliant
- Configurable number of endpoints from 1 to 8
- Up to 1024 bytes of dedicated packet buffer memory SRAM
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint support
- USB Suspend/Resume operations
- Frame locked clock pulse generation

The following additional feature is also available depending on the product implementation (see [Section 32.3: USB implementation](#)):

- USB 2.0 Link Power Management support

32.3 USB implementation

[Table 170](#) describes the USB implementation in the devices.

Table 170. STM32F302xx USB implementation

USB features ⁽¹⁾	STM32F302x6/8/D/E	STM32F302xB/C
	USB	
Number of endpoints	8	8
Size of dedicated packet buffer memory SRAM	1024 bytes ⁽²⁾	512 bytes ⁽³⁾
Dedicated packet buffer memory SRAM access scheme	2 x 16 bits / word	1 x 16 bits / word
USB 2.0 Link Power Management (LPM) support	X	N.A.

1. X= supported

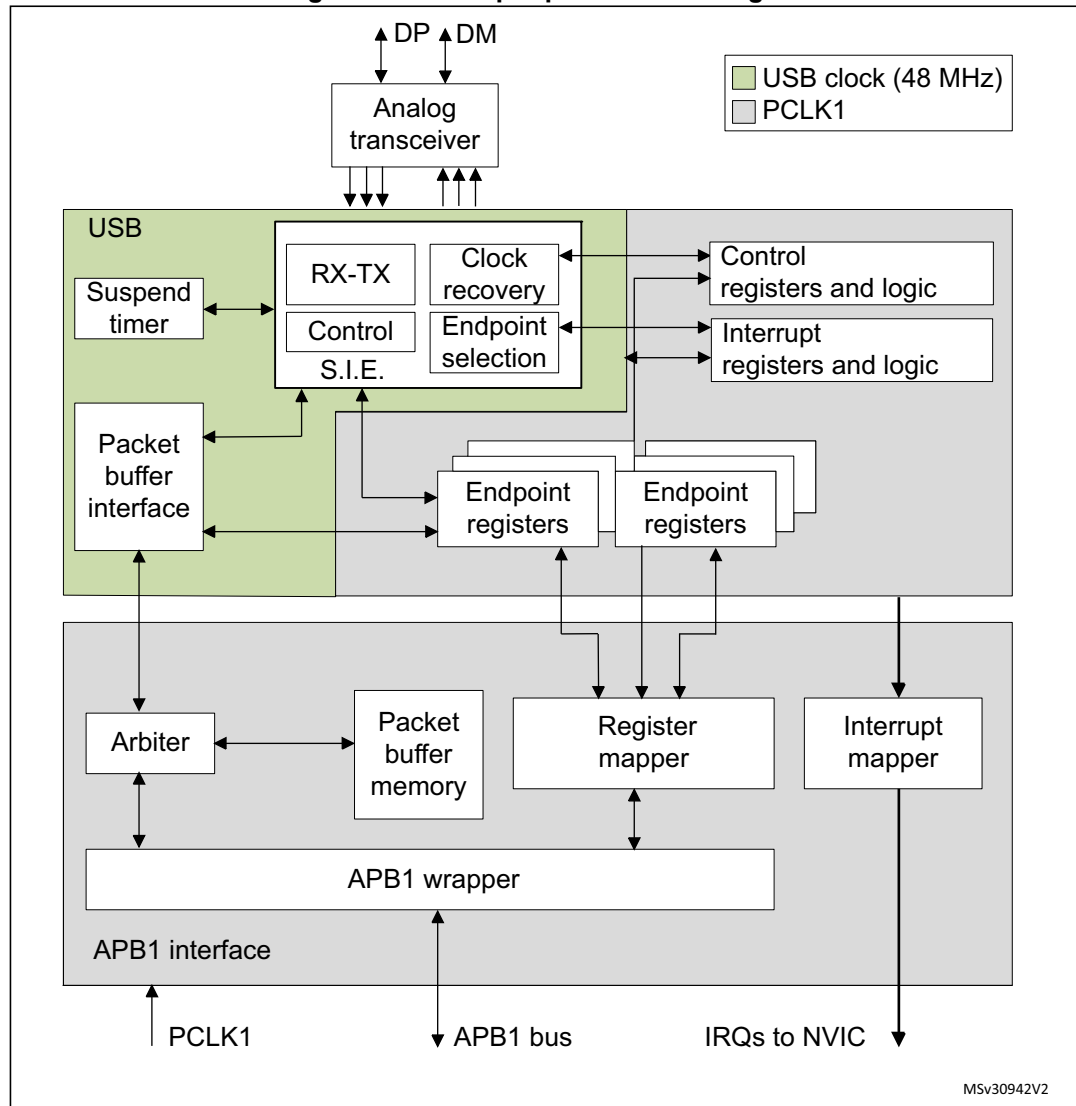
2. When the CAN peripheral clock is enabled in the RCC_APB1ENR register, only the first 768 Bytes are available to USB while the last 256 Bytes are used by CAN.

3. The 512 bytes are totally available to USB; nothing is shared with CAN.

32.4 USB functional description

Figure 390 shows the block diagram of the USB peripheral.

Figure 390. USB peripheral block diagram



The USB peripheral provides an USB-compliant connection between the host PC and the function implemented by the microcontroller. Data transfer between the host PC and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is up to 1024 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the USB host, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint is associated with a buffer description block indicating where the endpoint-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint is configured) takes place. The data

buffered by the USB peripheral is loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data has been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which allows to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wakeup line to allow the system to immediately restart the normal clock generation and/or support direct clock start/stop.

32.4.1 Description of USB blocks

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- **Serial Interface Engine (SIE):** The functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage,. This unit also generates signals according to USB peripheral events, such as Start of Frame (SOF), USB_Reset, Data errors etc. and to Endpoint related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.
- **Timer:** This block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- **Packet Buffer Interface:** This block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the Endpoint registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.
- **Endpoint-Related Registers:** Each endpoint has an associated register containing the endpoint type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer

endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.

- **Control Registers:** These are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- **Interrupt Registers:** These contain the Interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

Note: * Endpoint 0 is always used for control transfer in single-buffer mode.

The USB peripheral is connected to the APB1 bus through an APB1 interface, containing the following blocks:

- **Packet Memory:** This is the local memory that physically contains the Packet Buffers. It can be used by the Packet Buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the Packet Memory is up to 1024 bytes, structured as 512 half-words by 16 bits.
- **Arbiter:** This block accepts memory requests coming from the APB1 bus and from the USB interface. It resolves the conflicts by giving priority to APB1 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB1 transfers of any length are also allowed by this scheme.
- **Register Mapper:** This block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 16-bit wide half-word set addressed by the APB1.
- **APB1 Wrapper:** This provides an interface to the APB1 for the memory and register. It also maps the whole USB peripheral in the APB1 address space.
- **Interrupt Mapper:** This block is used to select how the possible USB events can generate interrupts and map them to three different lines of the NVIC:
 - USB low-priority interrupt (Channel 20): Triggered by all USB events (Correct transfer, USB reset, etc.). The firmware has to check the interrupt source before serving the interrupt.
 - USB high-priority interrupt (Channel 19): Triggered only by a correct transfer event for isochronous and double-buffer bulk transfer to reach the highest possible transfer rate.
 - USB wakeup interrupt (Channel 42): Triggered by the wakeup event from the USB Suspend mode.

32.5 Programming considerations

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

32.5.1 Generic USB device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and Isochronous transfers. Apart from system reset, action is always initiated by the USB peripheral, driven by one of the USB events described below.

32.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software should perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time (t_{STARTUP} specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the FRES bit in the CNTR register). Clearing the ISTR register then removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

USB reset (RESET interrupt)

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral will not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the Enable Function (EF) bit of the USB_DADDR register and initializing the EP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB_DADDR register, and configures any other necessary endpoint.

When a RESET interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of reset sequence which triggered the interrupt.

Structure and usage of packet buffers

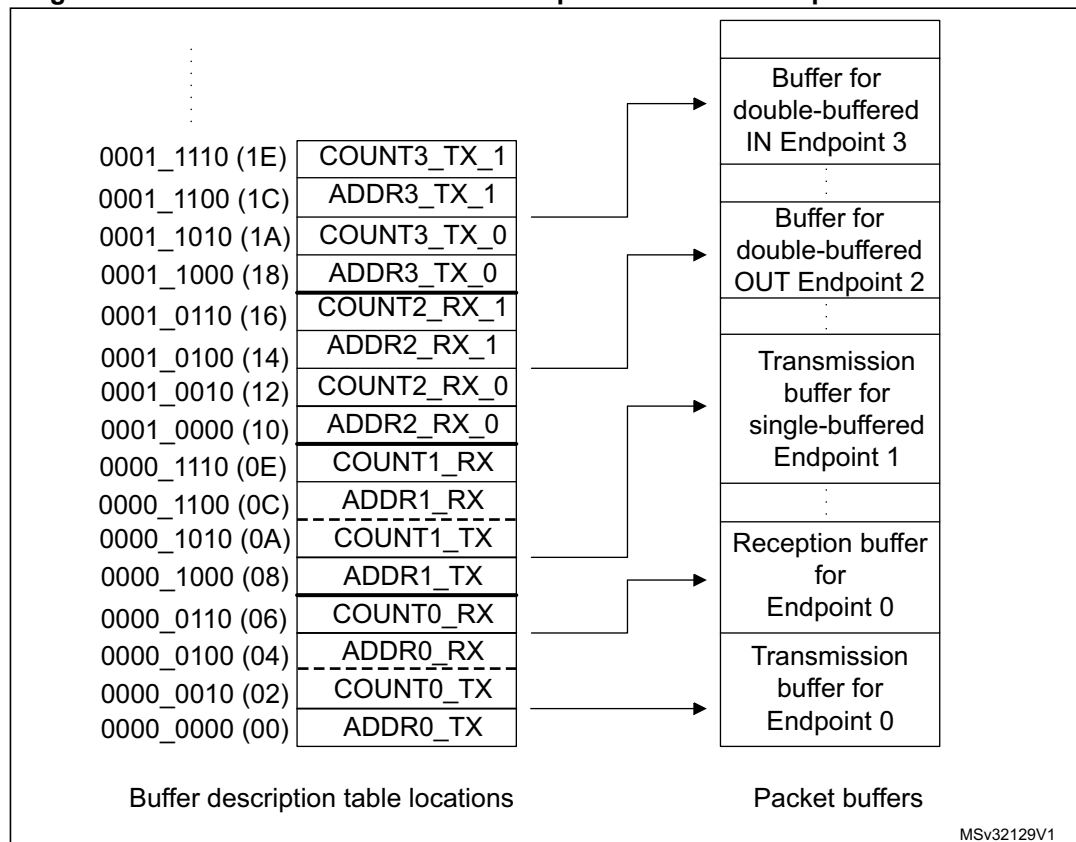
Each bidirectional endpoint may receive or transmit data from/to the host. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request and waits for its acknowledgment. Since the packet buffer memory has to be accessed by the microcontroller also, an arbitration logic takes care of the access conflicts, using half APB1 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both the agents can operate as if the packet memory is a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing

back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB1 bus. Different clock configurations are possible where the APB1 clock frequency can be higher or lower than the USB peripheral one.

Note: *Due to USB data rate and packet memory interface requirements, the APB1 clock must have a minimum frequency of 10 MHz to avoid data overrun/underrun problems.*

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory at the address indicated by the USB_BTABLE register. Each table entry is associated to an endpoint register and it is composed of four 16-bit half-words so that table start address must always be aligned to an 8-byte boundary (the lowest three bits of USB_BTABLE register are always “000”). Buffer descriptor table entries are described in the [Section 32.6.3: Buffer descriptor table](#). If an endpoint is unidirectional and it is neither an Isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to [Section 32.5.4: Isochronous transfers](#) and [Section 32.5.3: Double-buffered endpoints](#) respectively). The relationship between buffer description table entries and packet buffer areas is depicted in [Figure 391](#).

Figure 391. 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 will never change the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data will be copied to the memory only up to the last available location.

Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn_TX/ADDRn_RX registers so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The EP_TYPE bits in the USB_EPnR register must be set according to the endpoint type, eventually using the EP_KIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STAT_TX bits in the USB_EPnR register and COUNTn_TX must be initialized. For reception, STAT_RX bits must be set to enable reception and COUNTn_RX must be written with the allocated buffer size using the BL_SIZE and NUM_BLOCK fields. Unidirectional endpoints, except Isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB_EPnR and locations ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX (respectively), should not be modified by the application software, as the hardware can change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

IN packets (data transmission)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of ADDRn_TX and COUNTn_TX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16 bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (Refer to [Structure and usage of packet buffers on page 1010](#)) and starts sending a DATA0 or DATA1 PID according to USB_EPnR bit DTOG_TX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STAT_TX bits in the USB_EPnR register.

The ADDR internal register is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn_TX for COUNTn_TX/2 half-words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed will be used.

On receiving the ACK receipt by the host, the USB_EPnR register is updated in the following way: DTOG_TX bit is toggled, the endpoint is made invalid by setting STAT_TX=10 (NAK) and bit CTR_TX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP_ID and DIR bits in the USB_ISTR register. Servicing of the CTR_TX event starts clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STAT_TX to '11 (VALID) to re-enable transmissions. While the STAT_TX bits are equal to '10 (NAK), any IN request addressed to that endpoint is NAKed,

indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

OUT and SETUP packets (data reception)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn_RX and COUNTn_RX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn_RX is stored directly in its internal register ADDR. While COUNT is now reset and the values of BL_SIZE and NUM_BLOCK bit fields, which are read within COUNTn_RX content are used to initialize BUF_COUNT, an internal 16 bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but ACK packet is not sent and the ERR bit in USB_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STAT_RX in the USB_EPnR register and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, CRC included (i.e. data payload length + 2), or up to the last allocated memory location, as defined by BL_SIZE and NUM_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn_RX location inside the buffer description table entry, leaving unaffected BL_SIZE and NUM_BLOCK fields, which normally do not require to be re-written, and the USB_EPnR register is updated in the following way: DTOG_RX bit is toggled, the endpoint is made invalid by setting STAT_RX = '10 (NAK) and bit CTR_RX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP_ID and DIR bits in the USB_ISTR register. The CTR_RX event is serviced by first determining the transaction type (SETUP bit in the USB_EPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn_RX location inside the buffer description table entry related to the endpoint being

processed. After the received data is processed, the application software should set the STAT_RX bits to '11 (Valid) in the USB_EPnR, enabling further transactions. While the STAT_RX bits are equal to '10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

Control transfers

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOG_TX and DTOG_RX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STAT_TX and STAT_RX are set to '10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB_EPnR register at each CTR_RX event to distinguish normal OUT transactions from SETUP ones. A USB device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction should be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction should be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software will change NAK to VALID, otherwise to STALL. At the same time, if the status stage will be an OUT, the STATUS_OUT (EP_KIND in the USB_EPnR register) bit should be set, so that an error is generated if a status transaction is performed with not-zero data. When the status transaction is serviced, the application clears the STATUS_OUT bit and sets STAT_RX to VALID (to accept a new command) and STAT_TX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic doesn't allow a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STAT_RX bits are set to '01 (STALL) or '10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued CTR_RX request not yet acknowledged by the application (i.e. CTR_RX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the CTR_RX interrupt.

32.5.3 Double-buffered endpoints

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the

bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it will answer with a NAK handshake and the host PC will issue the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called 'double-buffering' can be used with bulk endpoints.

When 'double-buffering' is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both 'transmission' and 'reception' packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a 'reception' double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a 'transmission' double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB_EPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from '00 (Disabled): STAT_RX if the double-buffered bulk endpoint is enabled for reception, STAT_TX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB_EPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOG_RX (bit 14 of USB_EPnR register) for 'reception' double-buffered bulk endpoints or DTOG_TX (bit 6 of USB_EPnR register) for 'transmission' double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral should know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB_EPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW_BUF. In the following table the correspondence between USB_EPnR register bits and DTOG/SW_BUF definition is explained, for the cases of 'transmission' and 'reception' double-buffered bulk endpoints.

Table 171. Double-buffering buffer flag definition

Buffer flag	'Transmission' endpoint	'Reception' endpoint
DTOG	DTOG_TX (USB_EPnR bit 6)	DTOG_RX (USB_EPnR bit 14)
SW_BUF	USB_EPnR bit 14	USB_EPnR bit 6

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW_BUF

buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

Table 172. Bulk double-buffering memory buffers usage

Endpoint Type	DTOG	SW_BUF	Packet buffer used by USB Peripheral	Packet buffer used by Application Software
IN	0	1	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations.
	1	0	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations.	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	0	0	None ⁽¹⁾	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	1	1	None ⁽¹⁾	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
OUT	0	1	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.
	1	0	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	0	0	None ⁽¹⁾	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	1	1	None ⁽¹⁾	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by:

- Writing EP_TYPE bit field at '00 in its USB_EPnR register, to define the endpoint as a bulk, and
- Setting EP_KIND bit at '1 (DBL_BUF), in the same register.

The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the CTR_RX or CTR_TX bit of the addressed endpoint USB_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_EPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after DBL_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains '11 (Valid). However, as the token packet of a new transaction is received, the actual endpoint status will be masked as '10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW_BUF having the same value, see [Table 172 on page 1016](#)). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW_BUF bit, writing '1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by

the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control will take place and the actual transfer rate will be limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from '11 (Valid)' into the STAT bit pair of the related USB_EPnR register. In this case, the USB peripheral will always use the programmed endpoint status, regardless of the buffer usage condition.

32.5.4 Isochronous transfers

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as 'Isochronous'. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be 'isochronous' during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for Isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, Isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The Isochronous behavior for an endpoint is selected by setting the EP_TYPE bits at '10' in its USB_EPnR register; since there is no handshake phase the only legal values for the STAT_RX/STAT_TX bit pairs are '00 (Disabled)' and '11 (Valid)', any other value will produce results not compliant to USB standard. Isochronous endpoints implement double-buffering to ease application software development, using both 'transmission' and 'reception' packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOG_RX for 'reception' isochronous endpoints, DTOG_TX for 'transmission' isochronous endpoints, both in the related USB_EPnR register) according to [Table 173](#).

Table 173. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
IN	0	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.
	1	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.
OUT	0	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.
	1	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.

As it happens with double-buffered bulk endpoints, the USB_EPnR registers used to implement Isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have Isochronous endpoints enabled both for reception and transmission, two USB_EPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the CTR_RX or CTR_TX bit of the addressed endpoint USB_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_EPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for Isochronous transfers due to the lack of handshake phase, the endpoint remains always '11 (Valid). CRC errors or buffer-overflow conditions occurring during Isochronous OUT transfers are anyway considered as correct transactions and they always trigger an CTR_RX event. However, CRC errors will anyway set the ERR bit in the USB_ISTR register to notify the software of the possible data corruption.

32.5.5 Suspend/Resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to '1 in USB_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the FSUSP bit in the USB_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set LP_MODE bit in USB_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to insure that this process does not take more than 10 ms when the waking event is an USB reset sequence (See “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the LP_MODE bit in USB_CNTR register asynchronously. Even if this event can trigger an WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure should address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear FSUSP bit of USB_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB_FNR register can be used according to [Table 174](#), which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the Idle bus state; moreover at the end of a reset sequence the RESET bit in USB_ISTR register is set to 1, issuing an interrupt if enabled, which should be handled as usual.

Table 174. Resume event detection

[RXDP,RXDM] status	Wakeup event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (e.g. a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the RESUME bit in the USB_CNTR register to ‘1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOE interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the RESUME bit is clear, the resume sequence will be completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB_FNR register.

Note: *The RESUME bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the FSUSP bit in USB_CNTR register to 1.*

32.6 USB registers

The USB peripheral registers can be divided into the following groups:

- Common Registers: Interrupt and Control registers
- Endpoint Registers: Endpoint configuration and status
- Buffer Descriptor Table: Location of packet memory used to locate data buffers

All register addresses are expressed as offsets with respect to the USB peripheral registers base address 0x4000 5C00, except the buffer descriptor table locations, which starts at the address specified by the USB_BTABLER register. All register addresses are aligned to 32-bit word boundaries although they are 16-bit wide. On devices with “1 x 16 bits/word” access scheme, the same address alignment is used to access packet buffer memory locations, which are located starting from 0x4000 6000.

Refer to [Section 2.1 on page 42](#) for a list of abbreviations used in register descriptions.

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

32.6.1 Common registers

These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.

USB control register (USB_CNTR)

Address offset: 0x40

Reset value: 0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	L1REQM	Res	L1RESUME	RESUME	FSUSP	LP_MOD_E	PDWN	FRES
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bit 15 **CTRM**: Correct transfer interrupt mask

0: Correct Transfer (CTR) Interrupt disabled.

1: CTR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 14 **PMAOVRM**: Packet memory area over / underrun interrupt mask

0: PMAOVR Interrupt disabled.

1: PMAOVR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 13 **ERRM**: Error interrupt mask

0: ERR Interrupt disabled.

1: ERR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 12 **WKUPM**: Wakeup interrupt mask

0: WKUP Interrupt disabled.

1: WKUP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

- Bit 11 **SUSPM**: Suspend mode interrupt mask
0: Suspend Mode Request (SUSP) Interrupt disabled.
1: SUSP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 10 **RESETM**: USB reset interrupt mask
0: RESET Interrupt disabled.
1: RESET Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 9 **SOFM**: Start of frame interrupt mask
0: SOF Interrupt disabled.
1: SOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- Bit 8 **ESOFM**: Expected start of frame interrupt mask
0: Expected Start of Frame (ESOF) Interrupt disabled.
1: ESOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.
- 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.
Note: If LPM is not supported, this bit is not implemented and considered as reserved. Please refer to [Section 32.3: USB implementation](#).
- Bit 6 Reserved.
- Bit 5 **L1RESUME**: LPM L1 Resume request
The microcontroller can set this bit to send a LPM L1 Resume signal to the host. After the signalling ends, this bit is cleared by hardware.
Note: If LPM is not supported, this bit is not implemented and considered as reserved. Please refer to [Section 32.3: USB implementation](#).
- Bit 4 **RESUME**: Resume request
The microcontroller can set this bit to send a Resume signal to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the Host PC is ready to drive the resume sequence up to its end.
- Bit 3 **FSUSP**: Force suspend
Software must set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms.
0: No effect.
1: Enter suspend mode. Clocks and static power dissipation in the analog transceiver are left unaffected. If suspend power consumption is a requirement (bus-powered device), the application software should set the LP_MODE bit after FSUSP as explained below.

Bit 2 LP_MODE: Low-power mode

This mode is used when the suspend-mode power constraints require that all static power dissipation is avoided, except the one required to supply the external pull-up resistor. This condition should be entered when the application is ready to stop all system clocks, or reduce their frequency in order to meet the power consumption requirements of the USB suspend condition. The USB activity during the suspend mode (WKUP event) asynchronously resets this bit (it can also be reset by software).

0: No Low-power mode.

1: Enter Low-power mode.

Bit 1 PDWN: Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.

0: Exit Power Down.

1: Enter Power down mode.

Bit 0 FRES: Force USB Reset

0: Clear USB reset.

1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.

USB interrupt status register (USB_ISTR)

Address offset: 0x44

Reset value: 0x0000 0000

This register contains the status of all the interrupt sources allowing application software to

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Res.	Res.	Res.	DIR	EP_ID[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0				r	r	r	r	r

determine, which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, will perform all necessary actions, and finally it will clear the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line will be kept high again. If several bits are set simultaneously, only a single interrupt will be generated.

Endpoint transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_EPnR register (the CTR bit is actually a read only bit). For endpoint-related interrupts, the software can use the Direction of Transaction (DIR) and EP_ID read-only bits to identify, which endpoint made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB_ISTR events by specifying the order in which software checks USB_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt will be requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with '0 (these bits can only be cleared by software). Read-modify-write cycles should be avoided because between the read and the write operations another bit could be set by the hardware and the next write will clear it before the microprocessor has the time to serve the event.

The following describes each bit in detail:

Bit 15 CTR: Correct transfer

This bit is set by the hardware to indicate that an endpoint has successfully completed a transaction; using DIR and EP_ID bits software can determine which endpoint requested the interrupt. This bit is read-only.

Bit 14 PMAOVR: Packet memory area over / underrun

This bit is set if the microcontroller has not been able to respond in time to a USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host will retry the transaction. The PMAOVR interrupt should never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during Isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 13 ERR: Error

This flag is set whenever one of the errors listed below has occurred:

NANS: No ANSwer. The timeout for a host response has expired.

CRC: Cyclic Redundancy Check error. One of the received CRCs, either in the token or in the data, was wrong.

BST: Bit Stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.

FVIO: Framing format Violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).

The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (e.g. loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 12 WKUP: Wakeup

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the LP_MODE bit in the CTLR register and activates the USB_WAKEUP line, which can be used to notify the rest of the device (e.g. wakeup unit) about the start of the resume process. This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 11 SUSP: Suspend mode request

This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (FSUSP=1) until the end of resume sequence. This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 10 RESET: USB reset request

Set when the USB peripheral detects an active USB RESET signal at its inputs. The USB peripheral, in response to a RESET, just resets its internal protocol state machine, generating an interrupt if RESETM enable bit in the USB_CNTR register is set. Reception and transmission are disabled until the RESET bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RESET interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.

This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 9 SOF: Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine may monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB_FNR register which is updated at the SOF packet reception (this could be useful for isochronous applications).

This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 8 ESOF: Expected start of frame

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the hub does not receive it properly, the Suspend Timer issues this interrupt. If three consecutive ESOF interrupts are generated (i.e. three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the Suspend Timer is not yet locked. This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 7 L1REQ: LPM L1 state request

This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only '0 can be written and writing '1 has no effect.

Note: If LPM is not supported, this bit is not implemented and considered as reserved. Please refer to [Section 32.3: USB implementation](#).

Bits 6:5 Reserved.

Bit 4 **DIR**: Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.

If DIR bit=0, CTR_TX bit is set in the USB_EPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).

If DIR bit=1, CTR_RX bit or both CTR_TX/CTR_RX are set in the USB_EPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed.

This information can be used by the application software to access the USB_EPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **EP_ID[3:0]**: Endpoint Identifier

These bits are written by the hardware according to the endpoint number, which generated the interrupt request. If several endpoint transactions are pending, the hardware writes the endpoint identifier related to the endpoint having the highest priority defined in the following way: Two endpoint sets are defined, in order of priority: Isochronous and double-buffered bulk endpoints are considered first and then the other endpoints are examined. If more than one endpoint from the same set is requesting an interrupt, the EP_ID bits in USB_ISTR register are assigned according to the lowest requesting endpoint register, EP0R having the highest priority followed by EP1R and so on. The application software can assign a register to each endpoint according to this priority scheme, so as to order the concurring endpoint requests in a suitable way. These bits are read only.

USB frame number register (USB_FNR)

Address offset: 0x48

Reset value: 0x0XXX where X is undefined

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 15 **RXDP**: Receive data + line status

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wakeup event.

Bit 14 **RXDM**: Receive data - line status

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wakeup event.

Bit 13 **LCK**: Locked

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]**: Lost SOF

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]**: Frame number

This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for Isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

USB device address (USB_DADDR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved

Bit 7 **EF**: Enable function

This bit is set by the software to enable the USB device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at '0 no transactions are handled, irrespective of the settings of USB_EPnR registers.

Bits 6:0 **ADD[6:0]**: Device address

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the Endpoint Address (EA) field in the associated USB_EPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

Buffer table address (USB_BTABLE)

Address offset: 0x50

Reset value: 0x0000

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

Bits 15:3 **BTABLE[15:3]**: Buffer table

These bits contain the start address of the buffer allocation table inside the dedicated packet memory. This table describes each endpoint buffer location and size and it must be aligned to an 8 byte boundary (the 3 least significant bits are always '0). At the beginning of every transaction addressed to this device, the USB peripheral reads the element of this table related to the addressed endpoint, to get its buffer start location and the buffer size (Refer to [Structure and usage of packet buffers on page 1010](#)).

Bits 2:0 Reserved, forced by hardware to 0.

LPM control and status register (USB_LPMCSR)

Address offset: 0x54

Reset value: 0x0000

Note: *If LPM is not supported, this bit is not implemented and considered as reserved. Please refer to [Section 32.3: USB implementation](#).*

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		rw	rw

Bits 15:8 Reserved.

Bits 7:4 **BESL[3:0]**: BESL value

These bits contain the BESL value received with last ACKed LPM Token

Bit 3 **REM WAKE**: bRemoteWake value

This bit contains the bRemoteWake value received with last ACKed LPM Token

Bit 2 Reserved

Bit 1 **LPMACK**: LPM Token acknowledge enable

0: the valid LPM Token will be NYET.

1: the valid LPM Token will be ACK.

The NYET/ACK will be 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 **LP MEN**: LPM support enable

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.

32.6.2 Endpoint-specific registers

The number of these registers varies according to the number of endpoints that the USB peripheral is designed to handle. The USB peripheral supports up to 8 bidirectional endpoints. Each USB device must support a control endpoint whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint number value. For each endpoint, an USB_EPnR register is available to store the endpoint specific information.

USB endpoint n register (USB_EPnR), n=[0..7]

Address offset: 0x00 to 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

They are also reset when an USB reset is received from the USB bus or forced through bit FRES in the CTLR register, except the CTR_RX and CTR_TX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint has its USB_EPnR register where n is the endpoint identifier.

Read-modify-write cycles on these registers should be avoided because between the read and the write operations some bits could be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an 'invariant' value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their 'invariant' value.

Bit 15 CTR_RX: Correct Transfer for reception

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only '0' can be written, writing 1 has no effect.

Bit 14 DTOG_RX: Data Toggle, for reception transfers

If the endpoint is not Isochronous, this bit contains the expected value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent to the USB host, following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to [Section 32.5.3: Double-buffered endpoints](#)).

If the endpoint is Isochronous, this bit is used only to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to [Section 32.5.4: Isochronous transfers](#)). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes '0', the value of DTOG_RX remains unchanged, while writing '1' makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STAT_RX [1:0]**: Status bits, for reception transfers

These bits contain information about the endpoint status, which are listed in [Table 175: Reception status encoding on page 1031](#). These bits can be toggled by software to initialize their value. When the application software writes '0', the value remains unchanged, while writing '1' makes the bit value toggle. Hardware sets the STAT_RX bits to NAK when a correct transfer has occurred (CTR_RX=1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledge a new transaction

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to [Section 32.5.3: Double-buffered endpoints](#)).

If the endpoint is defined as Isochronous, its status can be only "VALID" or "DISABLED", so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT_RX bits to 'STALL' or 'NAK' for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing '1'.

Bit 11 **SETUP**: Setup transaction completed

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (CTR_RX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while CTR_RX bit is at 1; its state changes when CTR_RX is at 0. This bit is read-only.

Bits 10:9 **EP_TYPE[1:0]**: Endpoint type

These bits configure the behavior of this endpoint as described in [Table 176: Endpoint type encoding on page 1032](#). Endpoint 0 must always be a control endpoint and each USB function must have at least one control endpoint which has address 0, but there may be other control endpoints if required. Only control endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint is defined as NAK, the USB peripheral will not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint is defined as STALL in the receive direction, then the SETUP packet will be accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint is a control one.

Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EP_KIND configuration bit.

The usage of Isochronous endpoints is explained in [Section 32.5.4: Isochronous transfers](#)

Bit 8 **EP_KIND**: Endpoint kind

The meaning of this bit depends on the endpoint type configured by the EP_TYPE bits. [Table 177](#) 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 32.5.3: Double-buffered endpoints](#).

STATUS_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered 'STALL' instead of 'ACK'. This bit may be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 CTR_TX: Correct Transfer for transmission

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only '0' can be written.

Bit 6 DTOG_TX: Data Toggle, for transmission transfers

If the endpoint is non-isochronous, this bit contains the required value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to [Section 32.5.3: Double-buffered endpoints](#))

If the endpoint is Isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to [Section 32.5.4: Isochronous transfers](#)). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for Isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes '0', the value of DTOG_TX remains unchanged, while writing '1' makes the bit value toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 STAT_TX [1:0]: Status bits, for transmission transfers

These bits contain the information about the endpoint status, listed in [Table 178](#). These bits can be toggled by the software to initialize their value. When the application software writes '0', the value remains unchanged, while writing '1' makes the bit value toggle. Hardware sets the STAT_TX bits to NAK, when a correct transfer has occurred (CTR_TX=1) corresponding to a IN or SETUP (control only) transaction addressed to this endpoint. It then waits for the software to prepare the next set of data to be transmitted.

Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to [Section 32.5.3: Double-buffered endpoints](#)).

If the endpoint is defined as Isochronous, its status can only be "VALID" or "DISABLED".

Therefore, the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT_TX bits to 'STALL' or 'NAK' for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing '1'.

Bits 3:0 EA[3:0]: Endpoint address

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

Table 175. Reception status encoding

STAT_RX[1:0]	Meaning
00	DISABLED: all reception requests addressed to this endpoint are ignored.
01	STALL: the endpoint is stalled and all reception requests result in a STALL handshake.
10	NAK: the endpoint is naked and all reception requests result in a NAK handshake.
11	VALID: this endpoint is enabled for reception.

Table 176. Endpoint type encoding

EP_TYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

Table 177. Endpoint kind meaning

EP_TYPE[1:0]		EP_KIND meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	Not used
11	INTERRUPT	Not used

Table 178. Transmission status encoding

STAT_TX[1:0]	Meaning
00	DISABLED : all transmission requests addressed to this endpoint are ignored.
01	STALL : the endpoint is stalled and all transmission requests result in a STALL handshake.
10	NAK : the endpoint is naked and all transmission requests result in a NAK handshake.
11	VALID : this endpoint is enabled for transmission.

32.6.3 Buffer descriptor table

Although the buffer descriptor table is located inside the packet buffer memory, its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

On devices with “1 x 16 bits/word” access scheme, all packet memory locations are accessed by the APB using 32-bit aligned addresses, instead of the actual memory location addresses utilized by the USB peripheral for the USB_BTABLER register and buffer description table locations.

In the following pages, two address locations are reported for devices with “1 x 16 bits/word” access scheme: the one to be used by application software while accessing the packet memory, and the local one relative to USB peripheral access. To obtain the correct memory address value to be used in the application software while accessing the packet memory, the actual memory location address must be multiplied by two.

On devices with “2 x 16 bits/word” access scheme, the address location to be used by application software is the same as the local one relative to USB peripheral access. The packet memory on these devices should be accessed only by byte (8-bit) or half-word (16-bit) accesses. Word (32-bit) accesses are not allowed.

The first packet memory location is located at 0x4000 6000. The buffer descriptor table entry associated with the USB_EPnR registers is described below.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in [Structure and usage of packet buffers on page 1010](#).

Transmission buffer address n (USB_ADDRn_TX)

Address offset (“1 x 16 bits/word” access scheme): [USB_BTABLER] + n*16

Address offset (“2 x 16 bits/word” access scheme): [USB_BTABLER] + n*8

USB local address: [USB_BTABLER] + n*8

Note: *In case of double-buffered or isochronous endpoints in the IN direction, this address location is referred to as USB_ADDRn_TX_0.*

In case of double-buffered or isochronous endpoints in the OUT direction, this address location is used for USB_ADDRn_RX_0.

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

Bits 15:1 **ADDRn_TX[15:1]**: Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint associated with the USB_EPnR register at the next IN token addressed to it.

Bit 0 Must always be written as ‘0 since packet memory is half-word wide and all packet buffers must be half-word aligned.

Transmission byte count n (USB_COUNTn_TX)

Address offset ("1 x 16 bits/word" access scheme): [USB_BTABLE] + n*16 + 4

Address offset ("2 x 16 bits/word" access scheme): [USB_BTABLE] + n*8 + 2

USB local address: [USB_BTABLE] + n*8 + 2

Note: *In case of double-buffered or isochronous endpoints in the IN direction, this address location is referred to as USB_COUNTn_TX_0.*

In case of double-buffered or isochronous endpoints in the OUT direction, this address location is used for USB_COUNTn_RX_0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	COUNTn_TX[9:0]									
						r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:10 These bits are not used since packet size is limited by USB specifications to 1023 bytes. Their value is not considered by the USB peripheral.

Bits 9:0 **COUNTn_TX[9:0]**: Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint associated with the USB_EPnR register at the next IN token addressed to it.

Reception buffer address n (USB_ADDRn_RX)

Address offset ("1 x 16 bits/word" access scheme): [USB_BTABLE] + n*16 + 8

Address offset ("2 x 16 bits/word" access scheme): [USB_BTABLE] + n*8 + 4

USB local address: [USB_BTABLE] + n*8 + 4

Note: *In case of double-buffered or isochronous endpoints in the OUT direction, this address location is referred to as USB_ADDRn_RX_1.*

In case of double-buffered or isochronous endpoints in the IN direction, this address location is used for USB_ADDRn_TX_1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRn_RX[15:1]															-
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	-

Bits 15:1 **ADDRn_RX[15:1]**: Reception buffer address

These bits point to the starting address of the packet buffer, which will contain the data received by the endpoint associated with the USB_EPnR register at the next OUT/SETUP token addressed to it.

Bit 0 This bit must always be written as '0 since packet memory is half-word wide and all packet buffers must be half-word aligned.

Reception byte count n (USB_COUNTn_RX)

Address offset ("1 x 16 bits/word" access scheme): [USB_BTABLE] + n*16 + 12

Address offset ("2 x 16 bits/word" access scheme): [USB_BTABLE] + n*8 + 6

USB local address: [USB_BTABLE] + n*8 + 6

Note: *In case of double-buffered or isochronous endpoints in the OUT direction, this address location is referred to as USB_COUNTn_RX_1.*

In case of double-buffered or isochronous endpoints in the IN direction, this address location is used for USB_COUNTn_TX_1.

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

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (See "Universal Serial Bus Specification").

Bit 15 BL_SIZE: Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BL_SIZE=0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BL_SIZE=1, the memory block is 32-byte large, which allows to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 14:10 NUM_BLOCK[4:0]: Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BL_SIZE value as illustrated in [Table 179](#).

Bits 9:0 COUNTn_RX[9:0]: Reception byte count

These bits contain the number of bytes received by the endpoint associated with the USB_EPnR register during the last OUT/SETUP transaction addressed to it.

Table 179. Definition of allocated buffer memory

Value of NUM_BLOCK[4:0]	Memory allocated when BL_SIZE=0	Memory allocated when BL_SIZE=1
0 ('00000)	Not allowed	32 bytes
1 ('00001)	2 bytes	64 bytes
2 ('00010)	4 bytes	96 bytes
3 ('00011)	6 bytes	128 bytes
...
14 ('01110)	28 bytes	480 bytes
15 ('01111)	30 bytes	512 bytes
16 ('10000)	32 bytes	544 bytes
...
29 ('11101)	58 bytes	960 bytes
30 ('11110)	60 bytes	992 bytes
31 ('11111)	62 bytes	N/A

32.6.4 USB register map

The table below provides the USB register map and reset values.

Table 180. USB register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USB_EP0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USB_EP1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USB_EP2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USB_EP3R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USB_EP4R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USB_EP5R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USB_EP6R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	USB_EP7R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0x3F	Reserved																																
0x40	USB_CNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	L1REQM	Res.	L1RESUME	RESUME	FSUSP	LPMODE	PDWN	FRES
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	1	1
0x44	USB_ISTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	L1REQ	Res.	Res.	DIR	EP_ID[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x48	USB_FNR	Res.	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.	EF	ADD[6:0]								
	Reset value																							0	0	0	0	0	0	0	0	0	0

Table 180. 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
0x50	USB_BTABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BTABLE[15:3]												Res.	Res.	Res.	
	Reset value																	0	0	0	0	0	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.	BESL[3:0]				REMWAKE	Res.	Res.	Res.	
	Reset value																									0	0	0	0	0	0		0

Refer to [Section 3.2.2: Memory map and register boundary addresses](#) for the register boundary addresses.

33 Debug support (DBG)

33.1 Overview

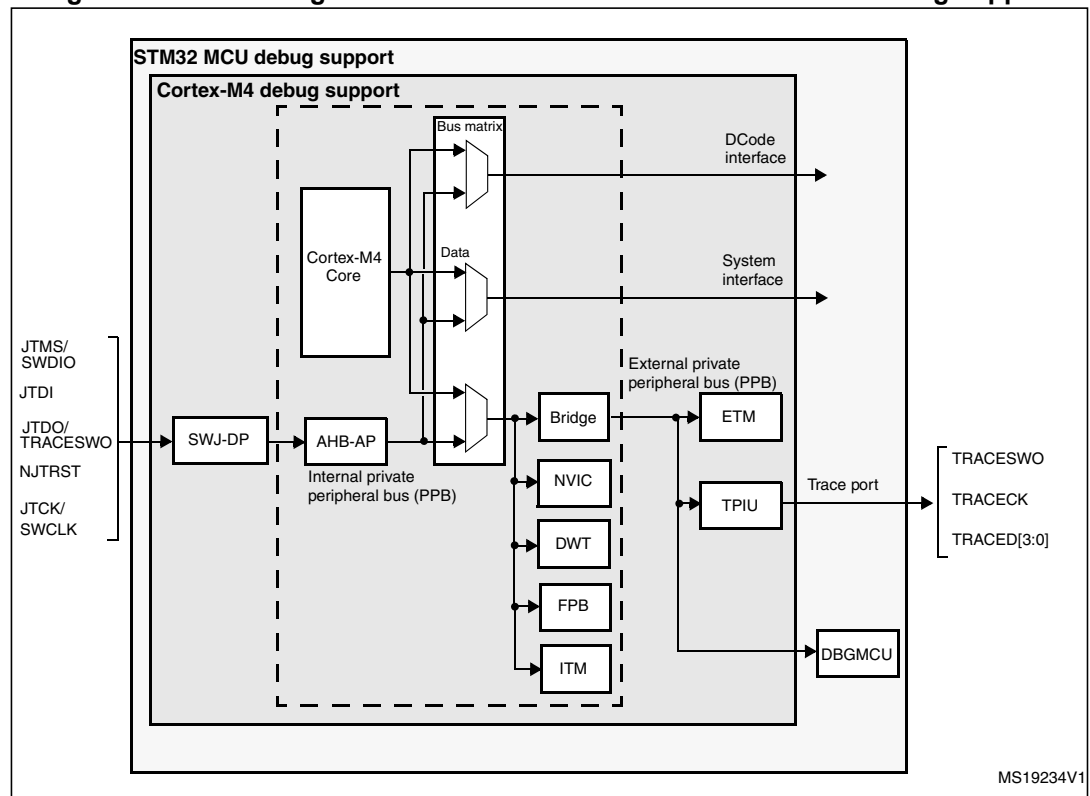
The STM32F302xx devices are built around a Cortex-M4[®]F core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and 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 STM32F302xx MCUs.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

Figure 392. Block diagram of STM32 MCU and Cortex-M4[®]F-level debug support



Note: The debug features embedded in the Cortex[®]-M4 core are a subset of the ARM[®] CoreSight Design Kit.

The ARM® Cortex-M4®F core provides integrated on-chip debug support. It is comprised of:

- SWJ-DP: Serial wire / JTAG debug port
- AHP-AP: AHB access port
- ITM: Instrumentation trace macrocell
- FPB: Flash patch breakpoint
- DWT: Data watchpoint trigger
- TPUI: Trace port unit interface (available on larger packages, where the corresponding pins are mapped)
- ETM: Embedded Trace Macrocell (available only on STM32F302xB/C devices larger packages, where the corresponding pins are mapped)

It also includes debug features dedicated to the STM32F302xx:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

Note: For further information on the debug feature supported by the ARM® Cortex-M4®F core, refer to the Cortex®-M4 with FPU-r0p1 Technical Reference Manual and to the CoreSight Design Kit-r0p1 TRM (see [Section 33.2: Reference ARM® documentation](#)).

33.2 Reference ARM® documentation

- Cortex-M4®F r0p1 Technical Reference Manual (TRM)
It is available from: <http://infocenter.arm.com>
- ARM® Debug Interface V5
- ARM® CoreSight Design Kit revision r0p1 Technical Reference Manual

33.3 SWJ debug port (serial wire and JTAG)

The STM32F302xx core integrates the Serial Wire / JTAG Debug Port (SWJ-DP). It is an ARM® standard CoreSight debug port that combines a JTAG-DP (5-pin) interface and a SW-DP (2-pin) interface.

- The JTAG Debug Port (JTAG-DP) provides a 5-pin standard JTAG interface to the AHP-AP port.
- The Serial Wire Debug Port (SW-DP) provides a 2-pin (clock + data) interface to the AHP-AP port.

In the SWJ-DP, the two JTAG pins of the SW-DP are multiplexed with some of the five JTAG pins of the JTAG-DP.

Figure 393. SWJ debug port

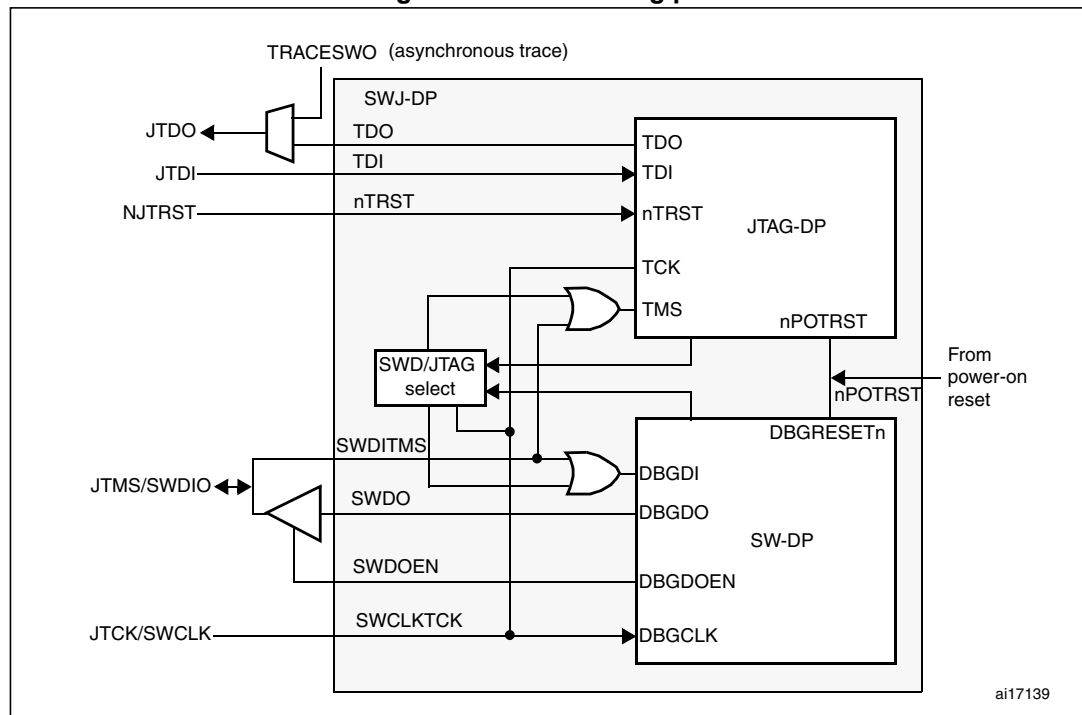


Figure 393 shows that the asynchronous TRACE output (TRACESWO) is multiplexed with TDO. This means that the asynchronous trace can only be used with SW-DP, not JTAG-DP.

33.3.1 Mechanism to select the JTAG-DP or the SW-DP

By default, the JTAG-Debug Port is active.

If the debugger host wants to switch to the SW-DP, it must provide a dedicated JTAG sequence on TMS/TCK (respectively mapped to SWDIO and SWCLK) which disables the JTAG-DP and enables the SW-DP. This way it is possible to activate the SWDP using only the SWCLK and SWDIO pins.

This sequence is:

1. Send more than 50 TCK cycles with TMS (SWDIO) = 1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111100111100111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) = 1

33.4 Pinout and debug port pins

The STM32F302xx MCUs are available in various packages with different numbers of available pins. As a result, some functionality (ETM) related to pin availability may differ between packages.

33.4.1 SWJ debug port pins

Five pins are used as outputs from the STM32F302xx for the SWJ-DP as *alternate functions* of general-purpose I/Os. These pins are available on all packages.

Table 181. SWJ debug port pins

SWJ-DP pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Debug assignment	
JTMS/SWDIO	I	JTAG Test Mode Selection	IO	Serial Wire Data Input/Output	PA13
JTCK/SWCLK	I	JTAG Test Clock	I	Serial Wire Clock	PA14
JTDI	I	JTAG Test Data Input	-	-	PA15
JTDO/TRACESWO	O	JTAG Test Data Output	-	TRACESWO if async trace is enabled	PB3
NJTRST	I	JTAG Test nReset	-	-	PB4

33.4.2 Flexible SWJ-DP pin assignment

After RESET (SYSRESETn or PORESETn), all five pins used for the SWJ-DP are assigned as dedicated pins immediately usable by the debugger host (note that the trace outputs are not assigned except if explicitly programmed by the debugger host).

However, it is possible to disable some or all of the SWJ-DP ports and so, to release the associated pins for general-purpose I/O(GPIO) usage. For more details on how to disable SWJ-DP port pins, please refer to [Section 11.3.2: I/O pin alternate function multiplexer and mapping](#).

Table 182. Flexible SWJ-DP pin assignment

Available debug ports	SWJ IO pin assigned				
	PA13 / JTMS / SWDIO	PA14 / JTCK / SWCLK	PA15 / JTDI	PB3 / JTDO	PB4 / NJTRST
Full SWJ (JTAG-DP + SW-DP) - Reset State	X	X	X	X	X
Full SWJ (JTAG-DP + SW-DP) but without NJTRST	X	X	X	X	
JTAG-DP Disabled and SW-DP Enabled	X	X			
JTAG-DP Disabled and SW-DP Disabled					Released

Note: When the APB bridge write buffer is full, it takes one extra APB cycle when writing the AFIO_MAPR register. This is because the deactivation of the JTAGSW pins is done in two cycles to guarantee a clean level on the nTRST and TCK input signals of the core.

- Cycle 1: the JTAGSW input signals to the core are tied to 1 or 0 (to 1 for nTRST, TDI and TMS, to 0 for TCK)
- Cycle 2: the GPIO controller takes the control signals of the SWJTAG IO pins (like controls of direction, pull-up/down, Schmitt trigger activation, etc.).

33.4.3 Internal pull-up and pull-down on JTAG pins

It is necessary to ensure that the JTAG input pins are not floating since they are directly connected to flip-flops to control the debug mode features. Special care must be taken with the SWCLK/TCK pin which is directly connected to the clock of some of these flip-flops.

To avoid any uncontrolled IO levels, the device embeds internal pull-ups and pull-downs on the JTAG input pins:

- NJTRST: Internal pull-up
- JTDI: Internal pull-up
- JTMS/SWDIO: Internal pull-up
- TCK/SWCLK: Internal pull-down

Once a JTAG IO is released by the user software, the GPIO controller takes control again. The reset states of the GPIO control registers put the I/Os in the equivalent state:

- NJTRST: Input pull-up
- JTDI: Input pull-up
- JTMS/SWDIO: Input pull-up
- JTCK/SWCLK: Input pull-down
- JTDO: Input floating

The software can then use these I/Os as standard GPIOs.

Note: The JTAG IEEE standard recommends to add pull-ups on TDI, TMS and nTRST but there is no special recommendation for TCK. However, for JTCK, the device needs an integrated pull-down.

Having embedded pull-ups and pull-downs removes the need to add external resistors.

33.4.4 Using serial wire and releasing the unused debug pins as GPIOs

To use the serial wire DP to release some GPIOs, the user software must change the GPIO (PA15, PB3 and PB4) configuration mode in the GPIO_MODER register. This releases PA15, PB3 and PB4 which now become available as GPIOs.

When debugging, the host performs the following actions:

- Under system reset, all SWJ pins are assigned (JTAG-DP + SW-DP).
- Under system reset, the debugger host sends the JTAG sequence to switch from the JTAG-DP to the SW-DP.
- Still under system reset, the debugger sets a breakpoint on vector reset.
- The system reset is released and the Core halts.
- All the debug communications from this point are done using the SW-DP. The other JTAG pins can then be reassigned as GPIOs by the user software.

Note: For user software designs, note that:

To release the debug pins, remember that they will be first configured either in input-pull-up (nTRST, TMS, TDI) or pull-down (TCK) or output tristate (TDO) for a certain duration after reset until the instant when the user software releases the pins.

When debug pins (JTAG or SW or TRACE) are mapped, changing the corresponding IO pin configuration in the IOPORT controller has no effect.

33.5 STM32F302xx JTAG TAP connection

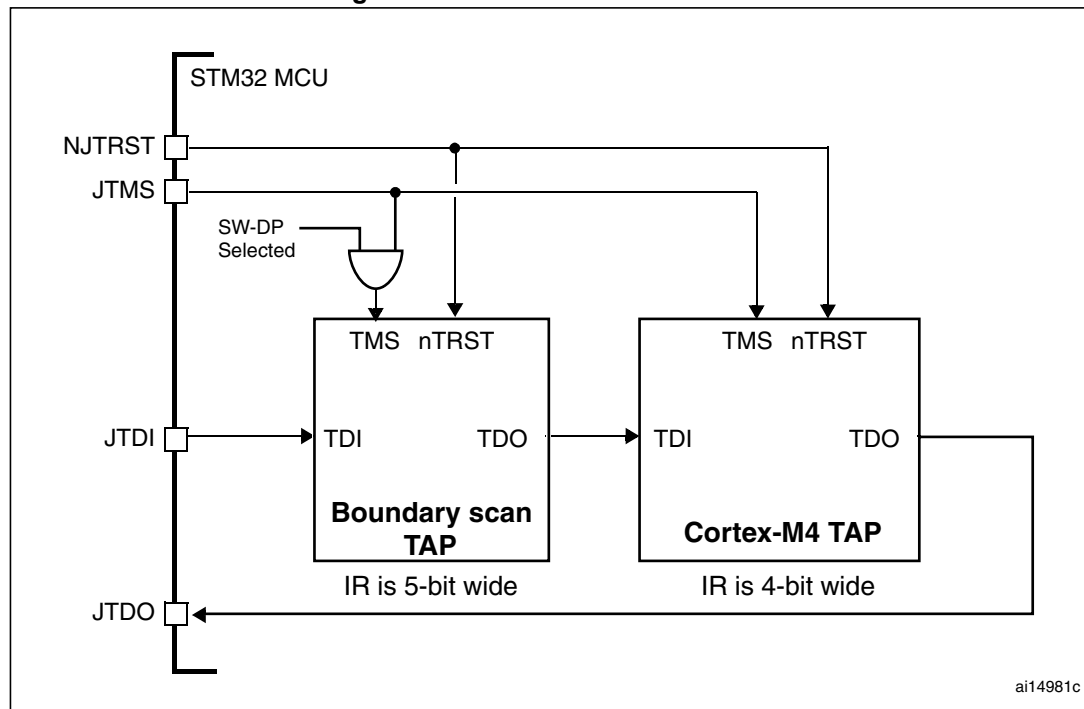
The STM32F302xx MCUs integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the Cortex-M4[®]F TAP (IR is 4-bit wide).

To access the TAP of the Cortex-M4[®]F for debug purposes:

1. First, it is necessary to shift the BYPASS instruction of the boundary scan TAP.
2. Then, for each IR shift, the scan chain contains 9 bits (=5+4) and the unused TAP instruction must be shifted in using the BYPASS instruction.
3. For each data shift, the unused TAP, which is in BYPASS mode, adds 1 extra data bit in the data scan chain.

Note: **Important:** Once Serial-Wire is selected using the dedicated ARM[®] JTAG sequence, the boundary scan TAP is automatically disabled (JTMS forced high).

Figure 394. JTAG TAP connections



33.6 ID codes and locking mechanism

There are several ID codes inside the STM32F302xx MCUs. ST strongly recommends tools designers to lock their debuggers using the MCU DEVICE ID code located in the external PPB memory map at address 0xE0042000.

33.6.1 MCU device ID code

The STM32F302xx MCUs integrate an MCU ID code. This ID identifies the ST MCU part-number and the die revision. It is part of the DBG_MCU component and is mapped on the external PPB bus (see [Section 33.14 on page 1055](#)). This code is accessible using the JTAG debug port (4 to 5 pins) or the SW debug port (two pins) or by the user software. It is even accessible while the MCU is under system reset.

DBGMCU_IDCODE

Address: 0xE004 2000

Only 32-bits access supported. Read-only

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

This code is read as 0x10000439 for Revision 1.0 of STM32F302x6x8 devices and 0x10000422 for Revision 1.0 of STM32F302xBxC devices.

Bits 31:16 **REV_ID(15:0)** Revision identifier

This field indicates the revision of the device. For example, it is read as 0x1000 for Revision 1.

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

Bits 11:0 **DEV_ID(11:0)**: Device identifier

This field indicates the device and its revision.

The device ID is 0x439 for STM32F302x6x8 devices, 0x422 for STM32F302xBxC devices and 0x446 for STM32F302xD/E.

33.6.2 Boundary scan TAP

JTAG ID code

The TAP of the STM32F302xx BSC (boundary scan) integrates a JTAG ID code equal to 0x06432041.

33.6.3 Cortex-M4[®]F TAP

The TAP of the ARM[®] Cortex-M4[®]F integrates a JTAG ID code. This ID code is the ARM[®] default one and has not been modified. This code is only accessible by the JTAG Debug Port.

This code is **0x4BA00477** (corresponds to Cortex-M4[®]F r0p1, see [Section 33.2: Reference ARM[®] documentation](#)).

Only the DEV_ID(11:0) should be used for identification by the debugger/programmer tools.

33.6.4 Cortex-M4[®]F JEDEC-106 ID code

The ARM[®] Cortex-M4[®]F integrates a JEDEC-106 ID code. It is located in the 4KB ROM table mapped on the internal PPB bus at address 0xE00FF000_0xE00FFFFF.

This code is accessible by the JTAG Debug Port (4 to 5 pins) or by the SW Debug Port (two pins) or by the user software.

33.7 JTAG debug port

A standard JTAG state machine is implemented with a 4-bit instruction register (IR) and five data registers (for full details, refer to the Cortex-M4[®]Fr0p1 *Technical Reference Manual (TRM)*, for references, please see [Section 33.2: Reference ARM[®] documentation](#)).

Table 183. JTAG debug port data registers

IR(3:0)	Data register	Details
1111	BYPASS [1 bit]	
1110	IDCODE [32 bits]	ID CODE 0x3BA00477 (ARM [®] Cortex-M4 [®] F r0p1 ID Code)
1010	DPACC [35 bits]	Debug port access register This initiates a debug port and allows access to a debug port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register. Bit 0 = RnW = Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved Refer to Table 184 for a description of the A(3:2) bits

Table 183. JTAG debug port data registers (continued)

IR(3:0)	Data register	Details
1011	APACC [35 bits]	<p>Access port access register</p> <p>Initiates an access port and allows access to an access port register.</p> <ul style="list-style-type: none"> When transferring data IN: <ul style="list-style-type: none"> Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request Bits 2:1 = A[3:2] = 2-bit address (sub-address AP registers). Bit 0 = RnW= Read request (1) or write request (0). When transferring data OUT: <ul style="list-style-type: none"> Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: <ul style="list-style-type: none"> 010 = OK/FAULT 001 = WAIT OTHER = reserved <p>There are many AP Registers (see AHB-AP) addressed as the combination of:</p> <ul style="list-style-type: none"> The shifted value A[3:2] The current value of the DP SELECT register
1000	ABORT [35 bits]	<p>Abort register</p> <ul style="list-style-type: none"> Bits 31:1 = Reserved Bit 0 = DAPABORT: write 1 to generate a DAP abort.

Table 184. 32-bit debug port registers addressed through the shifted value A[3:2]

Address	A(3:2) value	Description
0x0	00	Reserved, must be kept at reset value.
0x4	01	<p>DP CTRL/STAT register. Used to:</p> <ul style="list-style-type: none"> Request a system or debug power-up Configure the transfer operation for AP accesses Control the pushed compare and pushed verify operations. Read some status flags (overrun, power-up acknowledges)
0x8	10	<p>DP SELECT register: Used to select the current access port and the active 4-words register window.</p> <ul style="list-style-type: none"> 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	<p>DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)</p>

33.8 SW debug port

33.8.1 SW protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO bidirectional management, the line must be pulled-up on the board (100 K Ω recommended by ARM®).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

33.8.2 SW protocol sequence

Each sequence consist of three phases:

1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target
3. Data transfer phase (33 bits) transmitted by the host or the target

Table 185. Packet request (8-bits)

Bit	Name	Description
0	Start	Must be “1”
1	APnDP	0: DP Access 1: AP Access
2	RnW	0: Write Request 1: Read Request
4:3	A(3:2)	Address field of the DP or AP registers (refer to Table 184)
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-M4®F r0p1 *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 186. 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 187. 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.

33.8.3 SW-DP state machine (reset, idle states, ID code)

The State Machine of the SW-DP has an internal ID code which identifies the SW-DP. It follows the JEP-106 standard. This ID code is the default ARM® one and is set to **0x1BA01477** (corresponding to Cortex-M4®F r0p1).

Note: *Note that the SW-DP state machine is inactive until the target reads this ID code.*

- The SW-DP state machine is in RESET STATE either after power-on reset, or after the DP has switched from JTAG to SWD or after the line is high for more than 50 cycles
- The SW-DP state machine is in IDLE STATE if the line is low for at least two cycles after RESET state.
- After RESET state, it is **mandatory** to first enter into an IDLE state AND to perform a READ access of the DP-SW ID CODE register. Otherwise, the target will issue a FAULT acknowledge response on another transactions.

Further details of the SW-DP state machine can be found in the *Cortex-M4®F r0p1 TRM* and the *CoreSight Design Kit r0p1 TRM*.

33.8.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the target response can be immediate (if ACK=OK) or can be delayed (if ACK=WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result.
The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to know if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP or AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is "WAIT". With the exception of

IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.

- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state)
This is particularly important when writing the CTRL/STAT for a power-up request. If the next transaction (requiring a power-up) occurs immediately, it will fail.

33.8.5 SW-DP registers

Access to these registers are initiated when APnDP=0

Table 188. SW-DP registers

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
00	Read		IDCODE	The manufacturer code is not set to ST code 0x2BA01477 (identifies the SW-DP)
00	Write		ABORT	
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: <ul style="list-style-type: none"> – 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

33.8.6 SW-AP registers

Access to these registers are initiated when APnDP=1

There are many AP Registers (see AHB-AP) addressed as the combination of:

- The shifted value A[3:2]
- The current value of the DP SELECT register

33.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP

Features:

- System access is independent of the processor status.
- Either SW-DP or JTAG-DP accesses AHB-AP.
- The AHB-AP is an AHB master into the Bus Matrix. Consequently, it can access all the data buses (Dcode Bus, System Bus, internal and external PPB bus) but the ICode bus.
- Bitband transactions are supported.
- AHB-AP transactions bypass the FPB.

The address of the 32-bits AHB-AP registers are 6-bits wide (up to 64 words or 256 bytes) and consists of:

- d) Bits [7:4] = the bits [7:4] APBANKSEL of the DP SELECT register
- e) Bits [3:2] = the 2 address bits of A[3:2] of the 35-bit packet request for SW-DP.

The AHB-AP of the Cortex-M4[®]F includes 9 x 32-bits registers:

Table 189. Cortex-M4[®]F AHB-AP registers

Address offset	Register name	Notes
0x00	AHB-AP Control and Status Word	Configures and controls transfers through the AHB interface (size, hprot, status on current transfer, address increment type)
0x04	AHB-AP Transfer Address	
0x0C	AHB-AP Data Read/Write	
0x10	AHB-AP Banked Data 0	Directly maps the 4 aligned data words without rewriting the Transfer Address Register.
0x14	AHB-AP Banked Data 1	
0x18	AHB-AP Banked Data 2	
0x1C	AHB-AP Banked Data 3	
0xF8	AHB-AP Debug ROM Address	Base Address of the debug interface
0xFC	AHB-AP ID Register	

Refer to the *Cortex-M4[®]F r0p1 TRM* for further details.

33.10 Core debug

Core debug is accessed through the core debug registers. Debug access to these registers is by means of the *Advanced High-performance Bus* (AHB-AP) port. The processor can access these registers directly over the internal *Private Peripheral Bus* (PPB).

It consists of 4 registers:

Table 190. Core debug registers

Register	Description
DHCSR	The 32-bit Debug Halting Control and Status Register This provides status information about the state of the processor enable core debug halt and step the processor
DCRSR	The 17-bit Debug Core Register Selector Register: This selects the processor register to transfer data to or from.
DCRDR	The 32-bit Debug Core Register Data Register: This holds data for reading and writing registers to and from the processor selected by the DCRSR (Selector) register.
DEMCR	The 32-bit Debug Exception and Monitor Control Register: This provides Vector Catching and Debug Monitor Control. This register contains a bit named TRCENA which enable the use of a TRACE.

Note: **Important:** these registers are not reset by a system reset. They are only reset by a power-on reset.

Refer to the *Cortex-M4[®]F r0p1 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.

33.11 Capability of the debugger host to connect under system reset

The STM32F302xx MCUs' reset system comprises the following reset sources:

- POR (power-on reset) which asserts a RESET at each power-up.
- Internal watchdog reset
- Software reset
- External reset

The Cortex-M4[®]F differentiates the reset of the debug part (generally PORRESETn) and the other one (SYSRESETn)

This way, it is possible for the debugger to connect under System Reset, programming the Core Debug Registers to halt the core when fetching the reset vector. Then the host can release the system reset and the core will immediately halt without having executed any instructions. In addition, it is possible to program any debug features under System Reset.

Note: It is highly recommended for the debugger host to connect (set a breakpoint in the reset vector) under system reset.

33.12 FPB (Flash patch breakpoint)

The FPB unit:

- implements hardware breakpoints
- patches code and data from code space to system space. This feature gives the possibility to correct software bugs located in the Code Memory Space.

The use of a Software Patch or a Hardware Breakpoint is exclusive.

The FPB consists of:

- 2 literal comparators for matching against literal loads from Code Space and remapping to a corresponding area in the System Space.
- 6 instruction comparators for matching against instruction fetches from Code Space. They can be used either to remap to a corresponding area in the System Space or to generate a Breakpoint Instruction to the core.

33.13 DWT (data watchpoint trigger)

The DWT unit consists of four comparators. They are configurable as:

- a hardware watchpoint or
- a trigger to an ETM or
- a PC sampler or
- a data address sampler

The DWT also provides some means to give some profiling informations. For this, some counters are accessible to give the number of:

- Clock cycle
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- CPI (clock per instructions)
- Interrupt overhead

33.14 MCU debug component (DBGMCU)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog, I2C and bxCAN during a breakpoint
- Control of the trace pins assignment

33.14.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode, DBG_SLEEP bit of DBGMCU_CR register must be previously set by the debugger. This will feed HCLK with the same clock that is provided to FCLK (system clock previously configured by the software).
- In Stop mode, the bit DBG_STOP must be previously set by the debugger. This will enable the internal RC oscillator clock to feed FCLK and HCLK in STOP mode.

33.14.2 Debug support for timers, watchdog, bxCAN and I²C

During a breakpoint, it is necessary to choose how the counter of timers and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the bxCAN, the user can choose to block the update of the receive register during a breakpoint.

For the I²C, the user can choose to block the SMBUS timeout during a breakpoint.

33.14.3 Debug MCU configuration register

This register allows the configuration of the MCU under DEBUG. This concerns:

- Low-power mode support
- Timer and watchdog counter support
- bxCAN communication support
- Trace pin assignment

This DBGMCU_CR is mapped on the External PPB bus at address 0xE0042004.

It is asynchronously reset by the PORESET (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support these features, it is still possible for the user software to write to these registers.

DBGMCU_CR

Address: 0xE004 2004

Only 32-bit access supported

POR Reset: 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	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	TRACE MODE [1:0]		TRACE IOEN	Res.	Res	DBG STAND BY	DBG STOP	DBG SLEEP
								rw	rw	rw			rw	rw	rw

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

Bits 7:5 **TRACE_MODE[1:0] and TRACE_IOEN**: Trace pin assignment control

- With TRACE_IOEN=0:
 - TRACE_MODE=xx: TRACE pins not assigned (default state)
- With TRACE_IOEN=1:
 - TRACE_MODE=00: TRACE pin assignment for Asynchronous Mode
 - TRACE_MODE=01: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 1
 - TRACE_MODE=10: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 2
 - TRACE_MODE=11: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 4

Note: In STM32F302x6/8 devices, synchronous trace is not available, thus bits 7:5 are reserved and must be kept at 0.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Debug Standby mode

0: (FCLK=Off, HCLK=Off) The whole digital part is unpowered.

From software point of view, exiting from Standby is identical than fetching reset vector (except a few status bit indicated that the MCU is resuming from Standby)

1: (FCLK=On, HCLK=On) In this case, the digital part is not unpowered and FCLK and HCLK are provided by the internal RC oscillator which remains active. In addition, the MCU generate a system reset during Standby mode so that exiting from Standby is identical than fetching from reset

Bit 1 **DBG_STOP**: Debug Stop mode

0: (FCLK=Off, HCLK=Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI)). Consequently, the software must reprogram the clock controller to enable the PLL, the Xtal, etc.

1: (FCLK=On, HCLK=On) In this case, when entering STOP mode, FCLK and HCLK are provided by the internal RC oscillator which remains active in STOP mode. When exiting STOP mode, the software must reprogram the clock controller to enable the PLL, the Xtal, etc. (in the same way it would do in case of DBG_STOP=0)

Bit 0 **DBG_SLEEP**: Debug Sleep mode

0: (FCLK=On, HCLK=Off) In Sleep mode, FCLK is clocked by the system clock as previously configured by the software while HCLK is disabled.

In Sleep mode, the clock controller configuration is not reset and remains in the previously programmed state. Consequently, when exiting from Sleep mode, the software does not need to reconfigure the clock controller.

1: (FCLK=On, HCLK=On) In this case, when entering Sleep mode, HCLK is fed by the same clock that is provided to FCLK (system clock as previously configured by the software).

33.14.4 Debug MCU APB1 freeze register (DBGMCU_APB1_FZ)

The DBGMCU_APB1_FZ register is used to configure the MCU under DEBUG. It concerns the APB1 peripherals:

- Timer clock counter freeze
- I2C SMBUS timeout freeze
- Window watchdog and independent watchdog counter freeze support

This DBGMCU_APB1_FZ is mapped on the external PPB bus at address 0xE0042008.

The register is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address: 0xE004 2008

Only 32-bit access are supported.

Power on reset (POR): 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	DBG_I2C3_SMBUS_TIMEOUT ⁽¹⁾	Res	Res	Res	Res	DBG_CAN_STOP	Res	Res	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Res	Res	Res	Res	Res
						rw			rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res	Res	Res	Res	Res	DBG_TIM6_STOP	Res	DBG_TIM4_STOP ⁽²⁾	DBG_TIM3_STOP ⁽²⁾	DBG_TIM2_STOP
			rw	rw	rw						rw		rw	rw	rw

1. Only in STM32F302x6/8 and STM32F302xD/E devices.

2. Only in STM32F302xB/C/D/E devices.

Bits 31 Reserved, must be kept at reset value.

Bit 30 **DBG_I2C3_SMBUS_TIMEOUT**: SMBUS timeout mode stopped when core is halted

0: Same behavior as in normal mode

1: The SMBUS timeout is frozen

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **DBG_CAN_STOP**: Debug CAN stopped when core is halted

0: Same behavior as in normal mode

1: The CAN2 receive registers are frozen

Bits 24:23 Reserved, must be kept at reset value.

- Bit 22 **DBG_I2C2_SMBUS_TIMEOUT**: SMBUS timeout mode stopped when core is halted
0: Same behavior as in normal mode
1: The SMBUS timeout is frozen
- Bit 21 **DBG_I2C1_SMBUS_TIMEOUT**: SMBUS timeout mode stopped when core is halted
0: Same behavior as in normal mode
1: The SMBUS timeout is frozen
- Bits 20:13 Reserved, must be kept at reset value.
- Bit 12 **DBG_IWDG_STOP**: Debug independent watchdog stopped when core is halted
0: The independent watchdog counter clock continues even if the core is halted
1: The independent watchdog counter clock is stopped when the core is halted
- Bit 11 **DBG_WWDG_STOP**: Debug window watchdog stopped when core is halted
0: The window watchdog counter clock continues even if the core is halted
1: The window watchdog counter clock is stopped when the core is halted
- Bit 10 **DBG_RTC_STOP**: Debug RTC stopped when core is halted
0: The clock of the RTC counter is fed even if the core is halted
1: The clock of the RTC counter is stopped when the core is halted
- Bits 9:5 Reserved, must be kept at reset value.
- Bit 4 **DBG_TIM6_STOP**: TIM6 counter stopped when core is halted
0: The counter clock of TIM6 is fed even if the core is halted
1: The counter clock of TIM6 is stopped when the core is halted
- Bit 3 Reserved, must be kept at reset value.
- Bit 2 **DBG_TIM4_STOP**: TIM4 counter stopped when core is halted (Available on STM32F302xB/C devices only)
0: The counter clock of TIM4 is fed even if the core is halted
1: The counter clock of TIM4 is stopped when the core is halted
- Bit 1 **DBG_TIM3_STOP**: TIM3 counter stopped when core is halted
0: The counter clock of TIM3 is fed even if the core is halted
1: The counter clock of TIM3 is stopped when the core is halted
- Bit 0 **DBG_TIM2_STOP**: TIM2 counter stopped when core is halted
0: The counter clock of TIM2 is fed even if the core is halted
1: The counter clock of TIM2 is stopped when the core is halted

33.14.5 Debug MCU APB2 freeze register (DBGMCU_APB2_FZ)

The DBGMCU_APB2_FZ register is used to configure the MCU under DEBUG. It concerns APB2 peripherals:

- Timer clock counter freeze

This register is mapped on the external PPB bus at address 0xE004 200C

It is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address: 0xE004 200C

Only 32-bit access is supported.

POR: 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	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.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res	DBG_TIM1_STOP
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **DBG_TIMx_STOP**: TIMx counter stopped when core is halted (x=1, 8,15..17)

0: The clock of the involved timer counter is fed even if the core is halted

1: The clock of the involved timer counter is stopped when the core is halted

Note: Bit1 and Bit 5 are reserved.

33.15 TPIU (trace port interface unit)

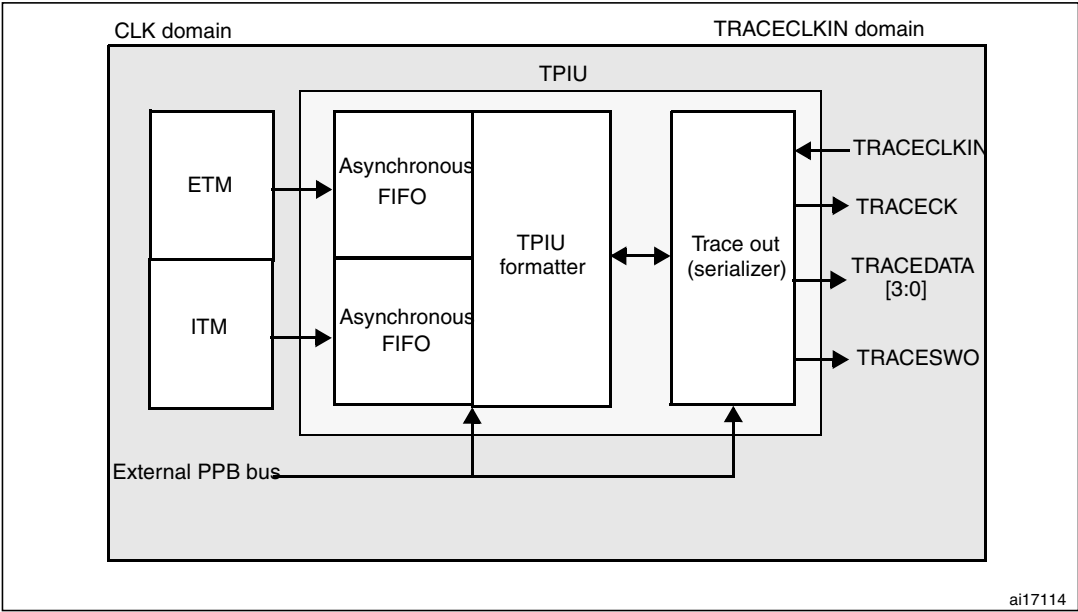
33.15.1 Introduction

The TPIU acts as a bridge between the on-chip trace data from the ITM and the ETM.

The output data stream encapsulates the trace source ID, that is then captured by a *trace port analyzer* (TPA).

The core embeds a simple TPIU, especially designed for low-cost debug (consisting of a special version of the CoreSight TPIU).

Figure 395. TPIU block diagram



33.15.2 TRACE pin assignment

- Asynchronous mode
The asynchronous mode requires 1 extra pin and is available on all packages. It is only available if using Serial Wire mode (not in JTAG mode).

Table 191. Asynchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32F302xx pin assignment
	Type	Description	
TRACESWO	O	TRACE Async Data Output	PB3

- Synchronous mode
The synchronous mode requires from 2 to 6 extra pins depending on the data trace size and is only available in the larger packages. In addition it is available in JTAG mode and in Serial Wire mode and provides better bandwidth output capabilities than asynchronous trace.

Table 192. Synchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32F302xx pin assignment
	Type	Description	
TRACECK	O	TRACE Clock	PE2
TRACED[3:0]	O	TRACE Sync Data Outputs Can be 1, 2 or 4.	PE[6:3]

TPUI TRACE pin assignment

By default, these pins are NOT assigned. They can be assigned by setting the TRACE_IOEN and TRACE_MODE bits in the **MCU Debug component configuration register**. This configuration has to be done by the debugger host.

In addition, the number of pins to assign depends on the trace configuration (asynchronous or synchronous).

- **Asynchronous mode:** 1 extra pin is needed
- **Synchronous mode** (Available in STM32F302xx only): from 2 to 5 extra pins are needed depending on the size of the data trace port register (1, 2 or 4) :
 - TRACECK
 - TRACED(0) if port size is configured to 1, 2 or 4
 - TRACED(1) if port size is configured to 2 or 4
 - TRACED(2) if port size is configured to 4
 - TRACED(3) if port size is configured to 4

To assign the TRACE pin, the debugger host must program the bits TRACE_IOEN and TRACE_MODE[1:0] of the Debug MCU configuration Register (DBGMCU_CR). By default the TRACE pins are not assigned.

This register is mapped on the external PPB and is reset by the PORESET (and not by the SYSTEM reset). It can be written by the debugger under SYSTEM reset.

Table 193. Flexible TRACE pin assignment

DBGMCU_CR register		Pins assigned for:	TRACE IO pin assigned					
TRACE_IOEN	TRACE_MODE [1:0]		PB3 / JTDO / TRACESWO	PE2 / TRACECK	PE3 / TRACED[0]	PE4 / TRACED[1]	PE5 / TRACED[2]	PE6 / TRACED[3]
0	XX	No Trace (default state)	Released ⁽¹⁾					
1	00	Asynchronous Trace	TRACESWO				Released (usable as GPIO)	
1	01	Synchronous Trace 1 bit	Released ⁽¹⁾	TRACECK	TRACED[0]			
1	10	Synchronous Trace 2 bit		TRACECK	TRACED[0]	TRACED[1]		
1	11	Synchronous Trace 4 bit		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]

1. When Serial Wire mode is used, it is released. But when JTAG is used, it is assigned to JTDO.

Note: By default, the TRACECLKIN input clock of the TPIU is tied to GND. It is assigned to HCLK two clock cycles after the bit TRACE_IOEN has been set.

Synchronous trace port availability depends on the chosen package.

The debugger must then program the Trace Mode by writing the PROTOCOL[1:0] bits in the SPP_R (Selected Pin Protocol) register of the TPIU.

- PROTOCOL=00: Trace Port Mode (synchronous)
- PROTOCOL=01 or 10: Serial Wire (Manchester or NRZ) Mode (asynchronous mode). Default state is 01

It then also configures the TRACE port size by writing the bits [3:0] in the CPSPS_R (Current Sync Port Size Register) of the TPIU:

- 0x1 for 1 pin (default state)
- 0x2 for 2 pins
- 0x8 for 4 pins

33.15.3 TPUI formatter

The formatter protocol outputs data in 16-byte frames:

- seven bytes of data
- eight bytes of mixed-use bytes consisting of:
 - 1 bit (LSB) to indicate it is a DATA byte ('0') or an ID byte ('1').
 - 7 bits (MSB) which can be data or change of source ID trace.
- one byte of auxiliary bits where each bit corresponds to one of the eight mixed-use bytes:
 - if the corresponding byte was a data, this bit gives bit0 of the data.
 - if the corresponding byte was an ID change, this bit indicates when that ID change takes effect.

Note: Refer to the ARM® CoreSight Architecture Specification v1.0 (ARM® IHI 0029B) for further information

33.15.4 TPUI frame synchronization packets

The TPUI can generate two types of synchronization packets:

- The Frame Synchronization packet (or Full Word Synchronization packet)
It consists of the word: 0x7F_FF_FF_FF (LSB emitted first). This sequence can not occur at any other time provided that the ID source code 0x7F has not been used.
It is output periodically **between** frames.
In continuous mode, the TPA must discard all these frames once a synchronization frame has been found.
- The Half-Word Synchronization packet
It consists of the half word: 0x7F_FF (LSB emitted first).
It is output periodically **between or within** frames.
These packets are only generated in continuous mode and enable the TPA to detect that the TRACE port is in IDLE mode (no TRACE to be captured). When detected by the TPA, it must be discarded.

33.15.5 Transmission of the synchronization frame packet

There is no Synchronization Counter register implemented in the TPIU of the core. Consequently, the synchronization trigger can only be generated by the **DWT**. Refer to the registers DWT Control Register (bits SYNCTAP[11:10]) and the DWT Current PC Sampler Cycle Count Register.

The TPUI Frame synchronization packet (0x7F_FF_FF_FF) is emitted:

- after each TPIU reset release. This reset is synchronously released with the rising edge of the TRACECLKIN clock. This means that this packet is transmitted when the TRACE_IOEN bit in the DBGMCU_CFG register is set. In this case, the word 0x7F_FF_FF_FF is not followed by any formatted packet.
- at each DWT trigger (assuming DWT has been previously configured). Two cases occur:
 - If the bit SYNENA of the ITM is reset, only the word 0x7F_FF_FF_FF is emitted without any formatted stream which follows.
 - If the bit SYNENA of the ITM is set, then the ITM synchronization packets will follow (0x80_00_00_00_00_00), formatted by the TPUI (trace source ID added).

33.15.6 Synchronous mode

The trace data output size can be configured to 4, 2 or 1 pin: TRACED(3:0)

The output clock is output to the debugger (TRACECK)

Here, TRACECLKIN is driven internally and is connected to HCLK only when TRACE is used.

Note: In this synchronous mode, it is not required to provide a stable clock frequency.

The TRACE I/Os (including TRACECK) are driven by the rising edge of TRACCLKIN (equal to HCLK). Consequently, the output frequency of TRACECK is equal to HCLK/2.

33.15.7 Asynchronous mode

This is a low-cost alternative to output the trace using only 1 pin: this is the asynchronous output pin TRACESWO. Obviously there is a limited bandwidth.

TRACESWO is multiplexed with JTDO when using the SW-DP pin. This way, this functionality is available in all STM32F302xx packages.

This asynchronous mode requires a constant frequency for TRACECLKIN. For the standard UART (NRZ) capture mechanism, 5% accuracy is needed. The Manchester encoded version is tolerant up to 10%.

33.15.8 TRACECLKIN connection inside the STM32F302xx

In the STM32F302xx, this TRACECLKIN input is internally connected to HCLK. This means that when in asynchronous trace mode, the application is restricted to use to time frames where the CPU frequency is stable.

Note: **Important:** when using asynchronous trace: it is important to be aware that:

The default clock of the STM32F302xx MCUs is the internal RC oscillator. Its frequency under reset is different from the one after reset release. This is because the RC calibration is the default one under system reset and is updated at each system reset release.

Consequently, the trace port analyzer (TPA) should not enable the trace (with the TRACE_IOEN bit) under system reset, because a Synchronization Frame Packet will be issued with a different bit time than trace packets which will be transmitted after reset release.

33.15.9 TPIU registers

The TPIU APB registers can be read and written only if the bit TRCENA of the Debug Exception and Monitor Control Register (DEMCR) is set. Otherwise, the registers are read as zero (the output of this bit enables the PCLK of the TPIU).

Table 194. Important TPIU registers

Address	Register	Description
0xE0040004	Current port size	Allows the trace port size to be selected: Bit 0: Port size = 1 Bit 1: Port size = 2 Bit 2: Port size = 3, not supported Bit 3: Port Size = 4 Only 1 bit must be set. By default, the port size is one bit. (0x00000001)
0xE00400F0	Selected pin protocol	Allows the Trace Port Protocol to be selected: Bit1:0= 00: Sync Trace Port Mode 01: Serial Wire Output - manchester (default value) 10: Serial Wire Output - NRZ 11: reserved
0xE0040304	Formatter and flush control	Bit 31-9 = always '0 Bit 8 = TrgIn = always '1 to indicate that triggers are indicated Bit 7-4 = always 0 Bit 3-2 = always 0 Bit 1 = EnFCont. In Sync Trace mode (Select_Pin_Protocol register bit1:0=00), this bit is forced to '1: the formatter is automatically enabled in continuous mode. In asynchronous mode (Select_Pin_Protocol register bit1:0 <> 00), this bit can be written to activate or not the formatter. Bit 0 = always 0 The resulting default value is 0x102 Note: In synchronous mode, because the TRACECTL pin is not mapped outside the chip, the formatter is always enabled in continuous mode -this way the formatter inserts some control packets to identify the source of the trace packets).
0xE0040300	Formatter and flush status	Not used in Cortex-M4®F, always read as 0x00000008

33.15.10 Example of configuration

- Set the bit TRCENA in the Debug Exception and Monitor Control Register (DEMCR)
- Write the TPIU Current Port Size Register to the desired value (default is 0x1 for a 1-bit port size)
- Write TPIU Formatter and Flush Control Register to 0x102 (default value)
- Write the TPIU Select Pin Protocol to select the sync or async mode. Example: 0x2 for async NRZ mode (UART like)
- Write the DBGMCU control register to 0x20 (bit IO_TRACEN) to assign TRACE I/Os for async mode. A TPIU Sync packet is emitted at this time (FF_FF_FF_7F)
- Configure the ITM and write the ITM Stimulus register to output a value

33.16 DBG register map

The following table summarizes the Debug registers

Table 195. DBG register map and reset values

	0xE004_200C	0xE004_2008	0xE004_2004	0xE004_2000	Addr.
	DBGMCU_APB2_FZ Reset value	DBGMCU_APB1_FZ Reset value	DBGMCU_CR Reset value	DBGMCU_IDCODE Reset value ⁽¹⁾	Register
	Res	Res	Res	X	31
	Res	DBG_I2C3_SMBUS_TIMEOUT	Res	X	30
	Res	Res	Res	X	29
	Res	Res	Res	X	28
	Res	Res	Res	X	27
	Res	Res	Res	X	26
	Res	DBG_CAN_STOP	Res	X	25
	Res	Res	Res	X	24
	Res	Res	Res	X	23
	Res	Res	Res	X	22
	Res	DBG_I2C2_SMBUS_TIMEOUT	Res	X	21
	Res	DBG_I2C1_SMBUS_TIMEOUT	Res	X	20
	Res	Res	Res	X	19
	Res	Res	Res	X	18
	Res	Res	Res	X	17
	Res	Res	Res	X	16
	Res	Res	Res	Res	15
	Res	Res	Res	Res	14
	Res	Res	Res	Res	13
	Res	DBG_IWDG_STOP	Res	Res	12
	Res	DBG_WWDG_STOP	Res	X	11
	Res	DBG_RTC_STOP	Res	X	10
	Res	Res	Res	X	9
	Res	Res	Res	X	8
	Res	Res	TRACE_MODE[1:0]	X	7
	Res	Res	0	X	6
	Res	Res	0	X	5
	Res	Res	0	X	4
	DBG_TIM17_STOP	0	TRACE_IOEN	X	3
	DBG_TIM16_STOP	Res	Res	X	2
	DBG_TIM15_STOP	0	DBG_STANDBY	X	1
	Res	0	DBG_STOP	X	0
	DBG_TIM1_STOP	0	DBG_SLEEP	X	

1. The reset value is product dependent. For more information, refer to [Section 33.6.1: MCU device ID code](#).

34 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 STM32F302xx microcontroller.

34.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x1FFF F7AC

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

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

Bits 31:0 **UID[31:0]**: X and Y coordinates on the wafer expressed in BCD format

Address offset: 0x04

Read only = 0XXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **UID[63:40]**: LOT_NUM[23:0]

Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]**: WAF_NUM[7:0]

Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0XXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]**: LOT_NUM[55:24]

Lot number (ASCII encoded)

34.2 Memory size data register

34.2.1 Flash size data register

Base address: 0x1FFF F7CC

Address offset: 0x00

Read only = 0XXXXX where X is factory-programmed

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

Bits 15:0 **FLASH_SIZE[15:0]**: Flash memory size

This bitfield indicates the size of the device Flash memory expressed in Kbytes.

As an example, 0x040 corresponds to 64 Kbytes.

35 Revision history

Table 196. Document revision history

Date	Revision	Changes
25-Apr-2014	1	Initial release.
16-May-2014	2	<p>Updated Section 9: System configuration controller (SYSCFG).</p> <p>Updated Section 25: Inter-integrated circuit (I2C) interface:</p> <p>unified the use of colors in Figure 251: Transfer bus diagrams for I2C slave transmitter, Figure 254: Transfer bus diagrams for I2C slave receiver, Figure 261: Transfer bus diagrams for I2C master transmitter, Figure 267: Transfer bus diagrams for SMBus slave transmitter (SBC=1), Figure 267: Transfer bus diagrams for SMBus slave transmitter (SBC=1) without changing their content.</p>
11-Sep-2014	3	<p>Overview of the manual</p> <ul style="list-style-type: none"> Updated Table 1: Available features related to each product <p>Analog-to-digital converters</p> <ul style="list-style-type: none"> Updated Figure 51: ADC block diagram, Figure 51: ADC block diagram, Figure 109: VBAT channel block diagram, Section 15.5.4: ADC configuration register (ADCx_CFGR, x=1..2), Section 15.5.18: ADC Analog Watchdog 2 Configuration Register (ADCx_AWD2CR, x=1..2), Section 15.5.19: ADC Analog Watchdog 3 Configuration Register (ADCx_AWD3CR, x=1..2), Section 15.5.20: ADC Differential Mode Selection Register (ADCx_DIFSEL, x=1..2) <p>Reset and Clock (RCC)</p> <ul style="list-style-type: none"> Updated Figure 36: STM32F302x6/8 clock tree Updated Section 10.4.13: Clock configuration register 3 (RCC_CFGR3), Section 10.4.10: Control/status register (RCC_CSR) and Section 10.4.14: RCC register map. <p>System configuration controller (SYSCFG)</p> <ul style="list-style-type: none"> Updated Section 12.1.6: SYSCFG configuration register 2 (SYSCFG_CFGR2) and Section 12.1.7: SYSCFG register map. <p>Interrupts and events</p> <ul style="list-style-type: none"> Updated Table 77: STM32F302xB/C/D/E vector table. <p>Operational amplifier (OPAMP)</p> <ul style="list-style-type: none"> Updated the Figure 120: STM32F302xB/C/D/E comparator and operational amplifier connections.

Table 196. Document revision history

Date	Revision	Changes
22-Jan-2015	4	<p>Added Section 9: Peripheral interconnect matrix. Extended the applicability to STM32F302xD/E devices. The following chapters were updated:</p> <p>Overview of the manual Updated Table 1: Available features related to each product</p> <p>System and memory overview Section 3.1: System architecture Added Table 3: STM32F302xD/E peripheral register boundary addresses Updated Section 3.3: Embedded SRAM</p> <p>Embedded Flash memory Section 4.1: Flash main features Section 4.2: Flash memory functional description Table 6: Flash module organization</p> <p>Option byte description Table 12: Option byte organization Table 13: Description of the option bytes</p> <p>Flexible memory controller (FMC) – Chapter added.</p> <p>Power control (PWR) – Section 8.1.1: Independent A/D and D/A converter supply and reference voltage</p> <p>Reset and clock control (RCC) Section 10.2: Clocks Section 10.4: RCC registers</p> <p>General purpose I/Os (GPIO) Section 11.2: GPIO main features Section 11.4: GPIO registers</p> <p>System configuration controller Section 12.1: SYSCFG registers</p> <p>Direct memory access controller (DMA) Table 73: STM32F302xB/C/D/E summary of DMA1 requests for each channel Table 75: STM32F302xB/C/D/E summary of DMA2 requests for each channel</p>

Table 196. Document revision history

Date	Revision	Changes
22-Jan-2015	4 (continued)	<p>Interrupts and events Table 77: STM32F302xB/C/D/E vector table</p> <p>Analog to digital converter (ADC) Section 15.2: ADC main features</p> <p>Digital to analog converter (DAC) Section 16.2: DAC1 main features Figure 111: DAC1 block diagram: updated the related note.</p> <p>Comparator (COMP) Section 17.2: COMP main features Section 17.5: COMP registers</p> <p>Operational amplifier (OPAMP) Section 18.2: OPAMP main features Section 18.4: OPAMP registers</p> <p>Advanced-control timers (TIM1) Section 20.1: TIM1 introduction</p> <p>General-purpose timers (TIM2/TIM3/TIM4) Section 21.1: TIM2/TIM3/TIM4 introduction</p> <p>Inter-integrated circuit (I2C) interface Section 28.3: I2C implementation</p> <p>Serial peripheral interface / inter-IC sound (SPI/I2S) Section 30.4: SPI/I2S implementation</p> <p>Universal synchronous asynchronous receiver transmitter (USART) Section 32.3: USB implementation</p> <p>Debug support (DBG) Section 33.14.4: Debug MCU APB1 freeze register (DBGMCU_APB1_FZ)</p>

Index

A

ADC_CDR	388
ADC_CR1	355, 357, 359, 362, 376, 380-382
ADC_CSR	383
ADC_HTR	368, 370
ADC_JDRx	380
ADC_JOFRx	379
ADC_JSQR	377
ADC_LTR	369
ADC_SMPR1	368
ADC_SMPR2	366
ADC_SQR1	371
ADC_SQR2	372
ADC_SQR3	374-375
ADCx_CCR, x=12	385

C

CAN_BTR	988
CAN_ESR	987
CAN_FA1R	999
CAN_FFA1R	999
CAN_FIRx	1000
CAN_FM1R	998
CAN_FMR	997
CAN_FS1R	998
CAN_IER	986
CAN_MCR	979
CAN_MSR	981
CAN_RDHxR	996
CAN_RDLxR	996
CAN_RDTxR	995
CAN_RF0R	984
CAN_RF1R	985
CAN_RlRx	994
CAN_TDHxR	993
CAN_TDLxR	993
CAN_TDTxR	992
CAN_TlRx	991
CAN_TSR	982
COMP_CSR	412-413, 417, 419
CRC_CR	85
CRC_DR	84
CRC_IDR	85
CRC_INIT	86
CRC_POL	86

D

DAC_CR	400
DAC_DHR12L1	403
DAC_DHR12R1	402
DAC_DHR8R1	403
DAC_DOR1	403
DAC_SR	404
DAC_SWTRIGR	402
DBGMCU_APB1_FZ	1058
DBGMCU_APB2_FZ	1060
DBGMCU_CR	1056
DBGMCU_IDCODE	1046
DMA_CCRx	260
DMA_CMARx	263
DMA_CNDTRx	262
DMA_CPARx	262
DMA_IFCR	259
DMA_ISR	258

E

EXTI_EMR	278, 281
EXTI_FTSR	279, 282
EXTI_IMR	278, 281
EXTI_PR	280, 283
EXTI_RTSTR	279, 282
EXTI_SWIER	280, 283

F

FLASH_ACR	72
FLASH_CR	74
FLASH_KEYR	72
FLASH_OPTKEYR	73
FLASH_SR	73
FSMC_BCR1..4	123
FSMC_BTR1..4	125
FSMC_BWTR1..4	128

G

GPIOx_AFRH	229
GPIOx_AFRL	229
GPIOx_BRR	230
GPIOx_BSRR	227
GPIOx_IDR	227
GPIOx_LCKR	228
GPIOx_MODER	225

GPIOx_ODR	227
GPIOx_OSPEEDR	226
GPIOx_OTYPER	225
GPIOx_PUPDR	226

I

I2C_ISR	829
I2Cx_CR1	819
I2Cx_CR2	822
I2Cx_ICR	831
I2Cx_OAR1	825
I2Cx_OAR2	826
I2Cx_PECR	832
I2Cx_RXDR	833
I2Cx_TIMEOUTR	828
I2Cx_TIMINGR	827
I2Cx_TXDR	833
IWDG_KR	716
IWDG_PR	717
IWDG_RLR	718
IWDG_SR	719
IWDG_WINR	720

O

OPAMP1_CSR	430
OPAMP2_CSR	432

P

purpose	616
PWR_CR	159
PWR_CSR	160

R

RCC_AHBENR	199
RCC_AHBSTR	208
RCC_APB1ENR	202
RCC_APB1RSTR	197
RCC_APB2ENR	201
RCC_APB2RSTR	195
RCC_BDCR	205
RCC_CFGR	189
RCC_CFGR2	209
RCC_CFGR3	211
RCC_CIR	193
RCC_CR	187
RCC_CSR	207
RTC_ALRMAR	750
RTC_ALRMBR	751
RTC_ALRMBSSR	763

RTC_BKxR	764
RTC_CALR	758
RTC_CR	742
RTC_DR	741
RTC_ISR	745
RTC_PRER	748
RTC_SHIFTR	754
RTC_SSR	753
RTC_TAFCR	759
RTC_TCR	759
RTC_TR	740
RTC_TSDR	756
RTC_TSSSR	757
RTC_TSTR	755
RTC_WPR	752
RTC_WUTR	749

S

SPI_CR1	947
SPI_CR2	949
SPI_CRCPR	954
SPI_DR	954
SPI_I2SCFGR	956
SPI_I2SPR	958
SPI_RXCR	955
SPI_TXCR	955
SPIx_CR2	949
SPIx_SR	952
SYSCFG_EXTICR1	235
SYSCFG_EXTICR2	236
SYSCFG_EXTICR3	238
SYSCFG_EXTICR4	240
SYSCFG_MEMRMP	233

T

TIM15_ARR	666
TIM15_BDTR	668
TIM15_CCER	662
TIM15_CCMR1	658
TIM15_CCR1	667
TIM15_CCR2	668
TIM15_CNT	665
TIM15_CR1	650
TIM15_CR2	651
TIM15_DCR	671
TIM15_DIER	655
TIM15_DMAR	671
TIM15_EGR	657
TIM15_PSC	666
TIM15_RCR	667
TIM15_SMCR	653

TIM15_SR 656
 TIM16_OR 690
 TIMx_ARR 532, 610, 705
 TIMx_BDTR 535, 687
 TIMx_CCER 528, 607, 682
 TIMx_CCMR1 522, 602, 680
 TIMx_CCMR2 527, 606
 TIMx_CCMR3 540
 TIMx_CCR1 533, 610, 686
 TIMx_CCR2 534, 611
 TIMx_CCR3 534, 611
 TIMx_CCR4 535, 612
 TIMx_CCR5 541
 TIMx_CCR6 542
 TIMx_CNT 532, 609, 684, 704
 TIMx_CR1 511, 592, 674, 701
 TIMx_CR2 512, 594, 675, 703
 TIMx_DCR 539, 613, 689
 TIMx_DIER 518, 598, 676, 703
 TIMx_DMAR 540, 613, 690
 TIMx_EGR 521, 601, 679, 704
 TIMx_OR 542
 TIMx_PSC 532, 610, 685, 705
 TIMx_RCR 533, 686
 TIMx_SMCR 515, 595
 TIMx_SR 519, 599, 678, 704
 TSC_CR 444
 TSC_ICR 447
 TSC_IER 446
 TSC_IOASCR 449
 TSC_IOCRR1 449
 TSC_IQGCSR 450
 TSC_IQXCR 450
 TSC_IQHCR 448
 TSC_IOSCR1 449
 TSC_ISR 448

U

USARTx_BRR 890
 USARTx_CR1 879
 USARTx_CR2 882
 USARTx_CR3 886
 USARTx_GTPR 890
 USARTx_ICR 898
 USARTx_ISR 893
 USARTx_RDR 899
 USARTx_RQR 892
 USARTx_RTOR 891
 USARTx_TDR 899
 USB_ADDRn_RX 1034
 USB_ADDRn_TX 1033

USB_BTABLE 1027
 USB_CNTR 1020, 1028
 USB_COUNTn_RX 1035
 USB_COUNTn_TX 1034
 USB_DADDR 1027
 USB_EPnR 1028
 USB_FNR 1025
 USBISTR 1022

W

WWDG_CFR 712
 WWDG_CR 711
 WWDG_SR 712

IMPORTANT NOTICE – PLEASE 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 acknowledgement.

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. 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.

© 2015 STMicroelectronics – All rights reserved

