



RM0090

Reference manual

STM32F405xx, STM32F407xx, STM32F415xx and STM32F417xx
advanced ARM-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32F405xx, STM32F407xx, STM32F415xx and STM32F417xx microcontroller memory and peripherals. The STM32F405xx, STM32F407xx, STM32F415xx and STM32F417xx will be referred to as STM32F40x and STM32F41x throughout the document, unless otherwise specified.

The STM32F40x and STM32F41x constitute a family of microcontrollers with different memory sizes, packages and peripherals.

For ordering information, mechanical and electrical device characteristics please refer to the STM32F40x and STM32F41x datasheets.

For information on programming, erasing and protection of the internal Flash memory please refer to the *STM32F40x and STM32F41x Flash programming manual*.

For information on the ARM Cortex™-M4F core, please refer to the *Cortex™-M4F Technical Reference Manual*.

Related documents

Available from www.arm.com:

- *Cortex™-M4F Technical Reference Manual, available from:*
http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439c/DDI0439C_cortex_m4_r0p1_trm.pdf

Available from STMicroelectronics web site (<http://www/st.com>):

- *STM32F40x and STM32F41x datasheets*
- *STM32F40x and STM32F41x Flash programming manual (PM0081)*

Contents

1	Documentation conventions	46
1.1	List of abbreviations for registers	46
1.2	Peripheral availability	46
2	Memory and bus architecture	47
2.1	System architecture	47
2.1.1	S0: I-bus	48
2.1.2	S1: D-bus	48
2.1.3	S2: S-bus	48
2.1.4	S3, S4: DMA memory bus	48
2.1.5	S5: DMA peripheral bus	49
2.1.6	S6: Ethernet DMA bus	49
2.1.7	S7: USB OTG HS DMA bus	49
2.1.8	BusMatrix	49
2.1.9	AHB/APB bridges (APB)	49
2.2	Memory organization	49
2.3	Memory map	50
2.3.1	Embedded SRAM	52
2.3.2	Bit banding	53
2.3.3	Embedded Flash memory	54
2.3.4	Flash memory read interface	54
2.3.5	Adaptive real-time memory accelerator (ART Accelerator™)	57
2.4	Boot configuration	57
3	CRC calculation unit	60
3.1	CRC introduction	60
3.2	CRC main features	60
3.3	CRC functional description	60
3.4	CRC registers	61
3.4.1	Data register (CRC_DR)	61
3.4.2	Independent data register (CRC_IDR)	61
3.4.3	Control register (CRC_CR)	62
3.4.4	CRC register map	62

4	Power control (PWR)	63
4.1	Power supplies	63
4.1.1	Independent A/D converter supply and reference voltage	64
4.1.2	Battery backup domain	65
4.1.3	Voltage regulator	67
4.2	Power supply supervisor	68
4.2.1	Power-on reset (POR)/power-down reset (PDR)	68
4.2.2	Brownout reset (BOR)	68
4.2.3	Programmable voltage detector (PVD)	69
4.3	Low-power modes	70
4.3.1	Slowing down system clocks	70
4.3.2	Peripheral clock gating	71
4.3.3	Sleep mode	71
4.3.4	Stop mode	72
4.3.5	Standby mode	73
4.3.6	Programming the RTC alternate functions to wake up the device from the Stop and Standby modes	75
4.4	Power control registers	78
4.4.1	PWR power control register (PWR_CR)	78
4.4.2	PWR power control/status register (PWR_CSR)	79
4.4.3	PWR register map	81
5	Reset and clock control (RCC)	82
5.1	Reset	82
5.1.1	System reset	82
5.1.2	Power reset	83
5.1.3	Backup domain reset	84
5.2	Clocks	84
5.2.1	HSE clock	86
5.2.2	HSI clock	87
5.2.3	PLL configuration	88
5.2.4	LSE clock	88
5.2.5	LSI clock	89
5.2.6	System clock (SYSCLK) selection	89
5.2.7	Clock security system (CSS)	89
5.2.8	RTC/AWU clock	90
5.2.9	Watchdog clock	90

5.2.10	Clock-out capability	91
5.2.11	Internal/external clock measurement using TIM5/TIM11	91
5.3	RCC registers	93
5.3.1	RCC clock control register (RCC_CR)	93
5.3.2	RCC PLL configuration register (RCC_PLLCFGGR)	95
5.3.3	RCC clock configuration register (RCC_CFGR)	97
5.3.4	RCC clock interrupt register (RCC_CIR)	99
5.3.5	RCC AHB1 peripheral reset register (RCC_AHB1RSTR)	102
5.3.6	RCC AHB2 peripheral reset register (RCC_AHB2RSTR)	104
5.3.7	RCC AHB3 peripheral reset register (RCC_AHB3RSTR)	105
5.3.8	RCC APB1 peripheral reset register (RCC_APB1RSTR)	105
5.3.9	RCC APB2 peripheral reset register (RCC_APB2RSTR)	108
5.3.10	RCC AHB1 peripheral clock register (RCC_AHB1ENR)	110
5.3.11	RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)	112
5.3.12	RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)	113
5.3.13	RCC APB1 peripheral clock enable register (RCC_APB1ENR)	113
5.3.14	RCC APB2 peripheral clock enable register (RCC_APB2ENR)	117
5.3.15	RCC AHB1 peripheral clock enable in low power mode register (RCC_AHB1LPENR)	119
5.3.16	RCC AHB2 peripheral clock enable in low power mode register (RCC_AHB2LPENR)	121
5.3.17	RCC AHB3 peripheral clock enable in low power mode register (RCC_AHB3LPENR)	122
5.3.18	RCC APB1 peripheral clock enable in low power mode register (RCC_APB1LPENR)	123
5.3.19	RCC APB2 peripheral clock enabled in low power mode register (RCC_APB2LPENR)	126
5.3.20	RCC Backup domain control register (RCC_BDCR)	128
5.3.21	RCC clock control & status register (RCC_CSR)	129
5.3.22	RCC spread spectrum clock generation register (RCC_SSCGR)	131
5.3.23	RCC PLLI2S configuration register (RCC_PLLI2SCFGR)	132
5.3.24	RCC register map	134
6	General-purpose I/Os (GPIO)	136
6.1	GPIO introduction	136
6.2	GPIO main features	136
6.3	GPIO functional description	136
6.3.1	General-purpose I/O (GPIO)	138

6.3.2	I/O pin multiplexer and mapping	139
6.3.3	I/O port control registers	141
6.3.4	I/O port data registers	141
6.3.5	I/O data bitwise handling	142
6.3.6	GPIO locking mechanism	142
6.3.7	I/O alternate function input/output	142
6.3.8	External interrupt/wakeup lines	143
6.3.9	Input configuration	143
6.3.10	Output configuration	144
6.3.11	Alternate function configuration	144
6.3.12	Analog configuration	145
6.3.13	Using the OSC32_IN/OSC32_OUT pins as GPIO PC14/PC15 port pins	146
6.3.14	Using the OSC_IN/OSC_OUT pins as GPIO PH0/PH1 port pins	146
6.3.15	Selection of RTC_AF1 and RTC_AF2 alternate functions	146
6.4	GPIO registers	148
6.4.1	GPIO port mode register (GPIOx_MODER) (x = A..I)	148
6.4.2	GPIO port output type register (GPIOx_OTYPER) (x = A..I)	148
6.4.3	GPIO port output speed register (GPIOx_OSPEEDR) (x = A..I)	149
6.4.4	GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..I)	149
6.4.5	GPIO port input data register (GPIOx_IDR) (x = A..I)	150
6.4.6	GPIO port output data register (GPIOx_ODR) (x = A..I)	150
6.4.7	GPIO port bit set/reset register (GPIOx_BSRR) (x = A..I)	150
6.4.8	GPIO port configuration lock register (GPIOx_LCKR) (x = A..I)	151
6.4.9	GPIO alternate function low register (GPIOx_AFRL) (x = A..I)	152
6.4.10	GPIO alternate function high register (GPIOx_AFRH) (x = A..I)	153
6.4.11	GPIO register map	153
7	System configuration controller (SYSCFG)	155
7.1	I/O compensation cell	155
7.2	SYSCFG registers	155
7.2.1	SYSCFG memory remap register (SYSCFG_MEMRMP)	155
7.2.2	SYSCFG peripheral mode configuration register (SYSCFG_PMC) ..	156
7.2.3	SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)	156

7.2.4	SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)	157
7.2.5	SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)	157
7.2.6	SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)	158
7.2.7	Compensation cell control register (SYSCFG_CMPCR)	158
7.2.8	SYSCFG register maps	159
8	DMA controller (DMA)	160
8.1	DMA introduction	160
8.2	DMA main features	161
8.3	DMA functional description	162
8.3.1	General description	162
8.3.2	DMA transactions	163
8.3.3	Channel selection	164
8.3.4	Arbiter	165
8.3.5	DMA streams	165
8.3.6	Source, destination and transfer modes	166
8.3.7	Pointer incrementation	169
8.3.8	Circular mode	170
8.3.9	Double buffer mode	170
8.3.10	Programmable data width, packing/unpacking, endianess	171
8.3.11	Single and burst transfers	173
8.3.12	FIFO	173
8.3.13	DMA transfer completion	176
8.3.14	DMA transfer suspension	177
8.3.15	Flow controller	177
8.3.16	Summary of the possible DMA configurations	178
8.3.17	Stream configuration procedure	179
8.3.18	Error management	180
8.4	DMA interrupts	181
8.5	DMA registers	181
8.5.1	DMA low interrupt status register (DMA_LISR)	181
8.5.2	DMA high interrupt status register (DMA_HISR)	182
8.5.3	DMA low interrupt flag clear register (DMA_LIFCR)	183
8.5.4	DMA high interrupt flag clear register (DMA_HIFCR)	184

8.5.5	DMA stream x configuration register (DMA_SxCR) (x = 0..7)	185
8.5.6	DMA stream x number of data register (DMA_SxNDTR) (x = 0..7)	188
8.5.7	DMA stream x peripheral address register (DMA_SxPAR) (x = 0..7)	188
8.5.8	DMA stream x memory 0 address register (DMA_SxM0AR) (x = 0..7)	188
8.5.9	DMA stream x memory 1 address register (DMA_SxM1AR) (x = 0..7)	189
8.5.10	DMA stream x FIFO control register (DMA_SxFCR) (x = 0..7)	190
8.5.11	DMA register map	191
9	Interrupts and events	195
9.1	Nested vectored interrupt controller (NVIC)	195
9.1.1	NVIC features	195
9.1.2	SysTick calibration value register	195
9.1.3	Interrupt and exception vectors	195
9.2	External interrupt/event controller (EXTI)	199
9.2.1	EXTI main features	199
9.2.2	EXTI block diagram	199
9.2.3	Wakeup event management	200
9.2.4	Functional description	200
9.2.5	External interrupt/event line mapping	201
9.3	EXTI registers	203
9.3.1	Interrupt mask register (EXTI_IMR)	203
9.3.2	Event mask register (EXTI_EMR)	203
9.3.3	Rising trigger selection register (EXTI_RTSR)	204
9.3.4	Falling trigger selection register (EXTI_FTSR)	204
9.3.5	Software interrupt event register (EXTI_SWIER)	205
9.3.6	Pending register (EXTI_PR)	205
9.3.7	EXTI register map	206
10	Analog-to-digital converter (ADC)	207
10.1	ADC introduction	207
10.2	ADC main features	207
10.3	ADC functional description	207
10.3.1	ADC on-off control	209
10.3.2	ADC clock	209
10.3.3	Channel selection	209
10.3.4	Single conversion mode	210
10.3.5	Continuous conversion mode	210

10.3.6	Timing diagram	211
10.3.7	Analog watchdog	211
10.3.8	Scan mode	212
10.3.9	Injected channel management	212
10.3.10	Discontinuous mode	213
10.4	Data alignment	214
10.5	Channel-wise programmable sampling time	215
10.6	Conversion on external trigger and trigger polarity	216
10.7	Fast conversion mode	217
10.8	Data management	218
10.8.1	Using the DMA	218
10.8.2	Managing a sequence of conversions without using the DMA	218
10.8.3	Conversions without DMA and without overrun detection	218
10.9	Multi ADC mode	219
10.9.1	Injected simultaneous mode	222
10.9.2	Regular simultaneous mode	223
10.9.3	Interleaved mode	225
10.9.4	Alternate trigger mode	226
10.9.5	Combined regular/injected simultaneous mode	228
10.9.6	Combined regular simultaneous + alternate trigger mode	228
10.10	Temperature sensor	229
10.11	Battery charge monitoring	230
10.12	ADC interrupts	231
10.13	ADC registers	232
10.13.1	ADC status register (ADC_SR)	232
10.13.2	ADC control register 1 (ADC_CR1)	233
10.13.3	ADC control register 2 (ADC_CR2)	235
10.13.4	ADC sample time register 1 (ADC_SMPR1)	238
10.13.5	ADC sample time register 2 (ADC_SMPR2)	238
10.13.6	ADC injected channel data offset register x (ADC_JOFRx)(x=1..4) ..	239
10.13.7	ADC watchdog higher threshold register (ADC_HTR)	239
10.13.8	ADC watchdog lower threshold register (ADC_LTR)	239
10.13.9	ADC regular sequence register 1 (ADC_SQR1)	240
10.13.10	ADC regular sequence register 2 (ADC_SQR2)	240
10.13.11	ADC regular sequence register 3 (ADC_SQR3)	241
10.13.12	ADC injected sequence register (ADC_JSQR)	241

10.13.13	ADC injected data register x (ADC_JDRx) (x= 1..4)	242
10.13.14	ADC regular data register (ADC_DR)	242
10.13.15	ADC Common status register (ADC_CSR)	244
10.13.16	ADC common control register (ADC_CCR)	245
10.13.17	ADC common regular data register for dual and triple modes (ADC_CDR)	247
10.13.18	ADC register map	247
11	Digital-to-analog converter (DAC)	250
11.1	DAC introduction	250
11.2	DAC main features	250
11.3	DAC functional description	252
11.3.1	DAC channel enable	252
11.3.2	DAC output buffer enable	252
11.3.3	DAC data format	252
11.3.4	DAC conversion	253
11.3.5	DAC output voltage	254
11.3.6	DAC trigger selection	254
11.3.7	DMA request	254
11.3.8	Noise generation	255
11.3.9	Triangle-wave generation	256
11.4	Dual DAC channel conversion	257
11.4.1	Independent trigger without wave generation	257
11.4.2	Independent trigger with single LFSR generation	258
11.4.3	Independent trigger with different LFSR generation	258
11.4.4	Independent trigger with single triangle generation	258
11.4.5	Independent trigger with different triangle generation	259
11.4.6	Simultaneous software start	259
11.4.7	Simultaneous trigger without wave generation	259
11.4.8	Simultaneous trigger with single LFSR generation	260
11.4.9	Simultaneous trigger with different LFSR generation	260
11.4.10	Simultaneous trigger with single triangle generation	260
11.4.11	Simultaneous trigger with different triangle generation	261
11.5	DAC registers	261
11.5.1	DAC control register (DAC_CR)	261
11.5.2	DAC software trigger register (DAC_SWTRIGR)	264

11.5.3	DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)	264
11.5.4	DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)	265
11.5.5	DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)	265
11.5.6	DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)	265
11.5.7	DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)	266
11.5.8	DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)	266
11.5.9	Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)	266
11.5.10	DUAL DAC 12-bit left aligned data holding register (DAC_DHR12LD)	267
11.5.11	DUAL DAC 8-bit right aligned data holding register (DAC_DHR8RD)	267
11.5.12	DAC channel1 data output register (DAC_DOR1)	268
11.5.13	DAC channel2 data output register (DAC_DOR2)	268
11.5.14	DAC status register (DAC_SR)	268
11.5.15	DAC register map	269
12	Digital camera interface (DCMI)	270
12.1	DCMI introduction	270
12.2	DCMI main features	270
12.3	DCMI pins	270
12.4	DCMI clocks	270
12.5	DCMI functional overview	271
12.5.1	DMA interface	272
12.5.2	DCMI physical interface	272
12.5.3	Synchronization	274
12.5.4	Capture modes	276
12.5.5	Crop feature	277
12.5.6	JPEG format	278
12.5.7	FIFO	278
12.6	Data format description	279
12.6.1	Data formats	279
12.6.2	Monochrome format	279

12.6.3	RGB format	279
12.6.4	YCbCr format	280
12.7	DCMI interrupts	280
12.8	DCMI register description	281
12.8.1	DCMI control register 1 (DCMI_CR)	281
12.8.2	DCMI status register (DCMI_SR)	283
12.8.3	DCMI raw interrupt status register (DCMI_RIS)	284
12.8.4	DCMI interrupt enable register (DCMI_IER)	285
12.8.5	DCMI masked interrupt status register (DCMI_MIS)	286
12.8.6	DCMI interrupt clear register (DCMI_ICR)	287
12.8.7	DCMI embedded synchronization code register (DCMI_ESCR)	287
12.8.8	DCMI embedded synchronization unmask register (DCMI_ESUR)	288
12.8.9	DCMI crop window start (DCMI_CWSTRT)	290
12.8.10	DCMI crop window size (DCMI_CWSIZE)	290
12.8.11	DCMI data register (DCMI_DR)	291
12.8.12	DCMI register map	291
13	Advanced-control timers (TIM1&TIM8)	293
13.1	TIM1&TIM8 introduction	293
13.2	TIM1&TIM8 main features	293
13.3	TIM1&TIM8 functional description	295
13.3.1	Time-base unit	295
13.3.2	Counter modes	296
13.3.3	Repetition counter	304
13.3.4	Clock selection	306
13.3.5	Capture/compare channels	309
13.3.6	Input capture mode	310
13.3.7	PWM input mode	311
13.3.8	Forced output mode	312
13.3.9	Output compare mode	313
13.3.10	PWM mode	314
13.3.11	Complementary outputs and dead-time insertion	317
13.3.12	Using the break function	318
13.3.13	Clearing the OCxREF signal on an external event	321
13.3.14	6-step PWM generation	322
13.3.15	One-pulse mode	323
13.3.16	Encoder interface mode	324

13.3.17	Timer input XOR function	327
13.3.18	Interfacing with Hall sensors	327
13.3.19	TIMx and external trigger synchronization	329
13.3.20	Timer synchronization	332
13.3.21	Debug mode	332
13.4	TIM1&TIM8 registers	333
13.4.1	TIM1&TIM8 control register 1 (TIMx_CR1)	333
13.4.2	TIM1&TIM8 control register 2 (TIMx_CR2)	334
13.4.3	TIM1&TIM8 slave mode control register (TIMx_SMCR)	337
13.4.4	TIM1&TIM8 DMA/interrupt enable register (TIMx_DIER)	339
13.4.5	TIM1&TIM8 status register (TIMx_SR)	341
13.4.6	TIM1&TIM8 event generation register (TIMx_EGR)	342
13.4.7	TIM1&TIM8 capture/compare mode register 1 (TIMx_CCMR1)	344
13.4.8	TIM1&TIM8 capture/compare mode register 2 (TIMx_CCMR2)	347
13.4.9	TIM1&TIM8 capture/compare enable register (TIMx_CCER)	348
13.4.10	TIM1&TIM8 counter (TIMx_CNT)	352
13.4.11	TIM1&TIM8 prescaler (TIMx_PSC)	352
13.4.12	TIM1&TIM8 auto-reload register (TIMx_ARR)	352
13.4.13	TIM1&TIM8 repetition counter register (TIMx_RCR)	353
13.4.14	TIM1&TIM8 capture/compare register 1 (TIMx_CCR1)	353
13.4.15	TIM1&TIM8 capture/compare register 2 (TIMx_CCR2)	354
13.4.16	TIM1&TIM8 capture/compare register 3 (TIMx_CCR3)	354
13.4.17	TIM1&TIM8 capture/compare register 4 (TIMx_CCR4)	355
13.4.18	TIM1&TIM8 break and dead-time register (TIMx_BDTR)	355
13.4.19	TIM1&TIM8 DMA control register (TIMx_DCR)	357
13.4.20	TIM1&TIM8 DMA address for full transfer (TIMx_DMAR)	358
13.4.21	TIM1&TIM8 register map	359
14	General-purpose timers (TIM2 to TIM5)	361
14.1	TIM2 to TIM5 introduction	361
14.2	TIM2 to TIM5 main features	361
14.3	TIM2 to TIM5 functional description	362
14.3.1	Time-base unit	362
14.3.2	Counter modes	363
14.3.3	Clock selection	372
14.3.4	Capture/compare channels	375
14.3.5	Input capture mode	376

14.3.6	PWM input mode	377
14.3.7	Forced output mode	378
14.3.8	Output compare mode	379
14.3.9	PWM mode	380
14.3.10	One-pulse mode	383
14.3.11	Clearing the OCxREF signal on an external event	384
14.3.12	Encoder interface mode	384
14.3.13	Timer input XOR function	387
14.3.14	Timers and external trigger synchronization	387
14.3.15	Timer synchronization	391
14.3.16	Debug mode	391
14.4	TIM2 to TIM5 registers	392
14.4.1	TIMx control register 1 (TIMx_CR1)	392
14.4.2	TIMx control register 2 (TIMx_CR2)	394
14.4.3	TIMx slave mode control register (TIMx_SMCR)	395
14.4.4	TIMx DMA/Interrupt enable register (TIMx_DIER)	396
14.4.5	TIMx status register (TIMx_SR)	397
14.4.6	TIMx event generation register (TIMx_EGR)	399
14.4.7	TIMx capture/compare mode register 1 (TIMx_CCMR1)	400
14.4.8	TIMx capture/compare mode register 2 (TIMx_CCMR2)	403
14.4.9	TIMx capture/compare enable register (TIMx_CCER)	404
14.4.10	TIMx counter (TIMx_CNT)	406
14.4.11	TIMx prescaler (TIMx_PSC)	406
14.4.12	TIMx auto-reload register (TIMx_ARR)	406
14.4.13	TIMx capture/compare register 1 (TIMx_CCR1)	407
14.4.14	TIMx capture/compare register 2 (TIMx_CCR2)	407
14.4.15	TIMx capture/compare register 3 (TIMx_CCR3)	408
14.4.16	TIMx capture/compare register 4 (TIMx_CCR4)	408
14.4.17	TIMx DMA control register (TIMx_DCR)	409
14.4.18	TIMx DMA address for full transfer (TIMx_DMAR)	409
14.4.19	TIM2 option register (TIM2_OR)	411
14.4.20	TIM5 option register (TIM5_OR)	412
14.4.21	TIMx register map	412
15	General-purpose timers (TIM9 to TIM14)	414
15.1	TIM9 to TIM14 introduction	414
15.2	TIM9 to TIM14 main features	414

15.2.1	TIM9/TIM12 main features	414
15.3	TIM10/TIM11 and TIM13/TIM14 main features	415
15.4	TIM9 to TIM14 functional description	417
15.4.1	Time-base unit	417
15.4.2	Counter modes	418
15.4.3	Clock selection	421
15.4.4	Capture/compare channels	423
15.4.5	Input capture mode	424
15.4.6	PWM input mode (only for TIM9/12)	425
15.4.7	Forced output mode	426
15.4.8	Output compare mode	427
15.4.9	PWM mode	428
15.4.10	One-pulse mode (only for TIM9/12)	429
15.4.11	TIM9/12 external trigger synchronization	430
15.4.12	Timer synchronization (TIM9/12)	433
15.4.13	Debug mode	433
15.5	TIM9 and TIM12 registers	434
15.5.1	TIM9/12 control register 1 (TIMx_CR1)	434
15.5.2	TIM9/12 control register 2 (TIMx_CR2)	435
15.5.3	TIM9/12 slave mode control register (TIMx_SMCR)	436
15.5.4	TIM9/12 Interrupt enable register (TIMx_DIER)	437
15.5.5	TIM9/12 status register (TIMx_SR)	438
15.5.6	TIM9/12 event generation register (TIMx_EGR)	439
15.5.7	TIM9/12 capture/compare mode register 1 (TIMx_CCMR1)	440
15.5.8	TIM9/12 capture/compare enable register (TIMx_CCER)	443
15.5.9	TIM9/12 counter (TIMx_CNT)	444
15.5.10	TIM9/12 prescaler (TIMx_PSC)	444
15.5.11	TIM9/12 auto-reload register (TIMx_ARR)	444
15.5.12	TIM9/12 capture/compare register 1 (TIMx_CCR1)	445
15.5.13	TIM9/12 capture/compare register 2 (TIMx_CCR2)	445
15.5.14	TIM9/12 register map	445
15.6	TIM10/11/13/14 registers	447
15.6.1	TIM10/11/13/14 control register 1 (TIMx_CR1)	447
15.6.2	TIM10/11/13/14 Interrupt enable register (TIMx_DIER)	448
15.6.3	TIM10/11/13/14 status register (TIMx_SR)	448
15.6.4	TIM10/11/13/14 event generation register (TIMx_EGR)	449

15.6.5	TIM10/11/13/14 capture/compare mode register 1 (TIMx_CCMR1)	450
15.6.6	TIM10/11/13/14 capture/compare enable register (TIMx_CCER)	452
15.6.7	TIM10/11/13/14 counter (TIMx_CNT)	453
15.6.8	TIM10/11/13/14 prescaler (TIMx_PSC)	453
15.6.9	TIM10/11/13/14 auto-reload register (TIMx_ARR)	453
15.6.10	TIM10/11/13/14 capture/compare register 1 (TIMx_CCR1)	454
15.6.11	TIM11 option register 1 (TIM11_OR)	454
15.6.12	TIM10/11/13/14 register map	455
16	Basic timers (TIM6&TIM7)	456
16.1	TIM6&TIM7 introduction	456
16.2	TIM6&TIM7 main features	456
16.3	TIM6&TIM7 functional description	457
16.3.1	Time-base unit	457
16.3.2	Counting mode	458
16.3.3	Clock source	461
16.3.4	Debug mode	461
16.4	TIM6&TIM7 registers	462
16.4.1	TIM6&TIM7 control register 1 (TIMx_CR1)	462
16.4.2	TIM6&TIM7 control register 2 (TIMx_CR2)	463
16.4.3	TIM6&TIM7 DMA/Interrupt enable register (TIMx_DIER)	463
16.4.4	TIM6&TIM7 status register (TIMx_SR)	464
16.4.5	TIM6&TIM7 event generation register (TIMx_EGR)	464
16.4.6	TIM6&TIM7 counter (TIMx_CNT)	464
16.4.7	TIM6&TIM7 prescaler (TIMx_PSC)	465
16.4.8	TIM6&TIM7 auto-reload register (TIMx_ARR)	465
16.4.9	TIM6&TIM7 register map	466
17	Independent watchdog (IWDG)	467
17.1	IWDG introduction	467
17.2	IWDG main features	467
17.3	IWDG functional description	467
17.3.1	Hardware watchdog	467
17.3.2	Register access protection	467
17.3.3	Debug mode	468

17.4	IWDG registers	468
17.4.1	Key register (IWDG_KR)	469
17.4.2	Prescaler register (IWDG_PR)	470
17.4.3	Reload register (IWDG_RLR)	470
17.4.4	Status register (IWDG_SR)	471
17.4.5	IWDG register map	471
18	Window watchdog (WWDG)	472
18.1	WWDG introduction	472
18.2	WWDG main features	472
18.3	WWDG functional description	472
18.4	How to program the watchdog timeout	474
18.5	Debug mode	475
18.6	WWDG registers	476
18.6.1	Control register (WWDG_CR)	476
18.6.2	Configuration register (WWDG_CFR)	477
18.6.3	Status register (WWDG_SR)	477
18.6.4	WWDG register map	478
19	Cryptographic processor (CRYP)	479
19.1	CRYP introduction	479
19.2	CRYP main features	479
19.3	CRYP functional description	480
19.3.1	DES/TDES cryptographic core	480
19.3.2	AES cryptographic core	485
19.3.3	Data type	492
19.3.4	Initialization vectors - CRYP_IV0...1(L/R)	494
19.3.5	CRYP busy state	496
19.3.6	Procedure to perform an encryption or a decryption	497
19.3.7	Context swapping	498
19.4	CRYP interrupts	499
19.5	CRYP DMA interface	500
19.6	CRYP registers	501
19.6.1	CRYP control register (CRYP_CR)	501
19.6.2	CRYP status register (CRYP_SR)	503
19.6.3	CRYP data input register (CRYP_DIN)	504

19.6.4	CRYP data output register (CRYP_DOUT)	505
19.6.5	CRYP DMA control register (CRYP_DMACR)	506
19.6.6	CRYP interrupt mask set/clear register (CRYP_IMSCR)	506
19.6.7	CRYP raw interrupt status register (CRYP_RISR)	507
19.6.8	CRYP masked interrupt status register (CRYP_MISR)	507
19.6.9	CRYP key registers (CRYP_K0...3(L/R)R)	508
19.6.10	CRYP initialization vector registers (CRYP_IV0...1(L/R)R)	510
19.6.11	CRYP register map	511
20	Random number generator (RNG)	513
20.1	RNG introduction	513
20.2	RNG main features	513
20.3	RNG functional description	513
20.3.1	Operation	514
20.3.2	Error management	514
20.4	RNG registers	514
20.4.1	RNG control register (RNG_CR)	515
20.4.2	RNG status register (RNG_SR)	515
20.4.3	RNG data register (RNG_DR)	516
20.4.4	RNG register map	517
21	Hash processor (HASH)	518
21.1	HASH introduction	518
21.2	HASH main features	518
21.3	HASH functional description	519
21.3.1	Duration of the processing	520
21.3.2	Data type	520
21.3.3	Message digest computing	521
21.3.4	Message padding	522
21.3.5	Hash operation	523
21.3.6	HMAC operation	524
21.3.7	Context swapping	524
21.3.8	HASH interrupt	526
21.4	HASH registers	526
21.4.1	HASH control register (HASH_CR)	527
21.4.2	HASH data input register (HASH_DIN)	529

21.4.3	HASH start register (HASH_STR)	530
21.4.4	HASH digest registers (HASH_HR0...4)	531
21.4.5	HASH interrupt enable register (HASH_IMR)	532
21.4.6	HASH status register (HASH_SR)	533
21.4.7	HASH context swap registers (HASH_CSR0...50)	534
21.4.8	HASH register map	535
22	Real-time clock (RTC)	536
22.1	Introduction	536
22.2	RTC main features	537
22.3	RTC functional description	538
22.3.1	Clock and prescalers	538
22.3.2	Real-time clock and calendar	539
22.3.3	Programmable alarms	539
22.3.4	Periodic auto-wakeup	540
22.3.5	RTC initialization and configuration	541
22.3.6	Reading the calendar	542
22.3.7	Resetting the RTC	543
22.3.8	RTC synchronization	544
22.3.9	RTC reference clock detection	544
22.3.10	RTC coarse digital calibration	545
22.3.11	RTC smooth digital calibration	546
22.3.12	Timestamp function	547
22.3.13	Tamper detection	548
22.3.14	Calibration clock output	550
22.3.15	Alarm output	550
22.4	RTC and low power modes	551
22.5	RTC interrupts	551
22.6	RTC registers	553
22.6.1	RTC time register (RTC_TR)	553
22.6.2	RTC date register (RTC_DR)	554
22.6.3	RTC control register (RTC_CR)	555
22.6.4	RTC initialization and status register (RTC_ISR)	557
22.6.5	RTC prescaler register (RTC_PRER)	559
22.6.6	RTC wakeup timer register (RTC_WUTR)	560
22.6.7	RTC calibration register (RTC_CALIBR)	561

22.6.8	RTC alarm A register (RTC_ALRMAR)	561
22.6.9	RTC alarm B register (RTC_ALRMBR)	562
22.6.10	RTC sub second register (RTC_SSR)	563
22.6.11	RTC shift control register (RTC_SHIFTR)	565
22.6.12	RTC write protection register (RTC_WPR)	566
22.6.13	RTC time stamp time register (RTC_TSTR)	566
22.6.14	RTC time stamp date register (RTC_TSDR)	567
22.6.15	RTC timestamp sub second register (RTC_TSSSR)	567
22.6.16	RTC calibration register (RTC_CALR)	568
22.6.17	RTC tamper and alternate function configuration register (RTC_TAFCR)	569
22.6.18	RTC alarm A sub second register (RTC_ALRMASSR)	571
22.6.19	RTC alarm B sub second register (RTC_ALRMBSSR)	572
22.6.20	RTC backup registers (RTC_BKPxR)	573
22.6.21	RTC register map	573
23	Inter-integrated circuit (I²C) interface	575
23.1	I ² C introduction	575
23.2	I ² C main features	575
23.3	I ² C functional description	576
23.3.1	Mode selection	576
23.3.2	I ² C slave mode	578
23.3.3	I ² C master mode	580
23.3.4	Error conditions	585
23.3.5	SDA/SCL line control	586
23.3.6	SMBus	586
23.3.7	DMA requests	589
23.3.8	Packet error checking	591
23.4	I ² C interrupts	591
23.5	I ² C debug mode	593
23.6	I ² C registers	593
23.6.1	I ² C Control register 1 (I2C_CR1)	593
23.6.2	I ² C Control register 2 (I2C_CR2)	595
23.6.3	I ² C Own address register 1 (I2C_OAR1)	597
23.6.4	I ² C Own address register 2 (I2C_OAR2)	597
23.6.5	I ² C Data register (I2C_DR)	598
23.6.6	I ² C Status register 1 (I2C_SR1)	598

23.6.7	I ² C Status register 2 (I2C_SR2)	602
23.6.8	I ² C Clock control register (I2C_CCR)	603
23.6.9	I ² C TRISE register (I2C_TRISE)	604
23.6.10	I ² C register map	605
24	Universal synchronous asynchronous receiver transmitter (USART)	606
24.1	USART introduction	606
24.2	USART main features	606
24.3	USART functional description	607
24.3.1	USART character description	610
24.3.2	Transmitter	611
24.3.3	Receiver	614
24.3.4	Fractional baud rate generation	619
24.3.5	USART receiver tolerance to clock deviation	628
24.3.6	Multiprocessor communication	629
24.3.7	Parity control	631
24.3.8	LIN (local interconnection network) mode	632
24.3.9	USART synchronous mode	634
24.3.10	Single-wire half-duplex communication	636
24.3.11	Smartcard	637
24.3.12	IrDA SIR ENDEC block	639
24.3.13	Continuous communication using DMA	641
24.3.14	Hardware flow control	643
24.4	USART interrupts	645
24.5	USART mode configuration	646
24.6	USART registers	646
24.6.1	Status register (USART_SR)	646
24.6.2	Data register (USART_DR)	648
24.6.3	Baud rate register (USART_BRR)	649
24.6.4	Control register 1 (USART_CR1)	649
24.6.5	Control register 2 (USART_CR2)	652
24.6.6	Control register 3 (USART_CR3)	653
24.6.7	Guard time and prescaler register (USART_GTPR)	656
24.6.8	USART register map	657
25	Serial peripheral interface (SPI)	658

25.1	SPI introduction	658
25.2	SPI and I ² S main features	659
25.2.1	SPI features	659
25.2.2	I ² S features	660
25.3	SPI functional description	661
25.3.1	General description	661
25.3.2	Configuring the SPI in slave mode	664
25.3.3	Configuring the SPI in master mode	667
25.3.4	Configuring the SPI for simplex communication	669
25.3.5	Data transmission and reception procedures	669
25.3.6	CRC calculation	676
25.3.7	Status flags	678
25.3.8	Disabling the SPI	679
25.3.9	SPI communication using DMA (direct memory addressing)	680
25.3.10	Error flags	682
25.3.11	SPI interrupts	683
25.4	I ² S functional description	684
25.4.1	I ² S general description	684
25.4.2	I ² S full duplex	685
25.4.3	Supported audio protocols	686
25.4.4	Clock generator	692
25.4.5	I ² S master mode	694
25.4.6	I ² S slave mode	696
25.4.7	Status flags	698
25.4.8	Error flags	699
25.4.9	I ² S interrupts	700
25.4.10	DMA features	700
25.5	SPI and I ² S registers	701
25.5.1	SPI control register 1 (SPI_CR1) (not used in I ² S mode)	701
25.5.2	SPI control register 2 (SPI_CR2)	703
25.5.3	SPI status register (SPI_SR)	704
25.5.4	SPI data register (SPI_DR)	705
25.5.5	SPI CRC polynomial register (SPI_CRCPR) (not used in I ² S mode)	705
25.5.6	SPI RX CRC register (SPI_RXCRCR) (not used in I ² S mode)	706
25.5.7	SPI TX CRC register (SPI_TXCRCR) (not used in I ² S mode)	706
25.5.8	SPI_I ² S configuration register (SPI_I2SCFGR)	707

25.5.9	SPI_I ² S prescaler register (SPI_I2SPR)	708
25.5.10	SPI register map	709
26	Secure digital input/output interface (SDIO)	710
26.1	SDIO main features	710
26.2	SDIO bus topology	710
26.3	SDIO functional description	712
26.3.1	SDIO adapter	714
26.3.2	SDIO APB2 interface	724
26.4	Card functional description	725
26.4.1	Card identification mode	725
26.4.2	Card reset	725
26.4.3	Operating voltage range validation	725
26.4.4	Card identification process	726
26.4.5	Block write	727
26.4.6	Block read	727
26.4.7	Stream access, stream write and stream read (MultiMediaCard only)	728
26.4.8	Erase: group erase and sector erase	729
26.4.9	Wide bus selection or deselection	730
26.4.10	Protection management	730
26.4.11	Card status register	733
26.4.12	SD status register	736
26.4.13	SD I/O mode	740
26.4.14	Commands and responses	741
26.5	Response formats	744
26.5.1	R1 (normal response command)	745
26.5.2	R1b	745
26.5.3	R2 (CID, CSD register)	745
26.5.4	R3 (OCR register)	746
26.5.5	R4 (Fast I/O)	746
26.5.6	R4b	746
26.5.7	R5 (interrupt request)	747
26.5.8	R6	748
26.6	SDIO I/O card-specific operations	748
26.6.1	SDIO I/O read wait operation by SDIO_D2 signalling	748
26.6.2	SDIO read wait operation by stopping SDIO_CK	749

26.6.3	SDIO suspend/resume operation	749
26.6.4	SDIO interrupts	749
26.7	CE-ATA specific operations	749
26.7.1	Command completion signal disable	749
26.7.2	Command completion signal enable	750
26.7.3	CE-ATA interrupt	750
26.7.4	Aborting CMD61	750
26.8	HW flow control	750
26.9	SDIO registers	751
26.9.1	SDIO power control register (SDIO_POWER)	751
26.9.2	SDI clock control register (SDIO_CLKCR)	751
26.9.3	SDIO argument register (SDIO_ARG)	752
26.9.4	SDIO command register (SDIO_CMD)	753
26.9.5	SDIO command response register (SDIO_RESPCMD)	754
26.9.6	SDIO response 1..4 register (SDIO_RESPx)	754
26.9.7	SDIO data timer register (SDIO_DTIMER)	755
26.9.8	SDIO data length register (SDIO_DLEN)	755
26.9.9	SDIO data control register (SDIO_DCTRL)	756
26.9.10	SDIO data counter register (SDIO_DCOUNT)	757
26.9.11	SDIO status register (SDIO_STA)	758
26.9.12	SDIO interrupt clear register (SDIO_ICR)	759
26.9.13	SDIO mask register (SDIO_MASK)	761
26.9.14	SDIO FIFO counter register (SDIO_FIFOCNT)	763
26.9.15	SDIO data FIFO register (SDIO_FIFO)	764
26.9.16	SDIO register map	764
27	Controller area network (bxCAN)	766
27.1	bxCAN introduction	766
27.2	bxCAN main features	766
27.3	bxCAN general description	767
27.3.1	CAN 2.0B active core	767
27.3.2	Control, status and configuration registers	767
27.3.3	Tx mailboxes	767
27.3.4	Acceptance filters	768
27.4	bxCAN operating modes	769
27.4.1	Initialization mode	770

27.4.2	Normal mode	770
27.4.3	Sleep mode (low power)	770
27.5	Test mode	771
27.5.1	Silent mode	771
27.5.2	Loop back mode	772
27.5.3	Loop back combined with silent mode	772
27.6	STM32F40x and STM32F41x in Debug mode	773
27.7	bxCAN functional description	773
27.7.1	Transmission handling	773
27.7.2	Time triggered communication mode	775
27.7.3	Reception handling	775
27.7.4	Identifier filtering	776
27.7.5	Message storage	780
27.7.6	Error management	782
27.7.7	Bit timing	782
27.8	bxCAN interrupts	785
27.9	CAN registers	786
27.9.1	Register access protection	786
27.9.2	CAN control and status registers	786
27.9.3	CAN mailbox registers	797
27.9.4	CAN filter registers	804
27.9.5	bxCAN register map	808
28	Ethernet (ETH): media access control (MAC) with DMA controller	811
28.1	Ethernet introduction	811
28.2	Ethernet main features	811
28.2.1	MAC core features	812
28.2.2	DMA features	813
28.2.3	PTP features	813
28.3	Ethernet pins	814
28.4	Ethernet functional description: SMI, MII and RMII	815
28.4.1	Station management interface: SMI	815
28.4.2	Media-independent interface: MII	818
28.4.3	Reduced media-independent interface: RMII	820
28.4.4	MII/RMII selection	821

28.5	Ethernet functional description: MAC 802.3	822
28.5.1	MAC 802.3 frame format	823
28.5.2	MAC frame transmission	826
28.5.3	MAC frame reception	833
28.5.4	MAC interrupts	838
28.5.5	MAC filtering	839
28.5.6	MAC loopback mode	842
28.5.7	MAC management counters: MMC	842
28.5.8	Power management: PMT	843
28.5.9	Precision time protocol (IEEE1588 PTP)	846
28.6	Ethernet functional description: DMA controller operation	852
28.6.1	Initialization of a transfer using DMA	853
28.6.2	Host bus burst access	853
28.6.3	Host data buffer alignment	854
28.6.4	Buffer size calculations	854
28.6.5	DMA arbiter	855
28.6.6	Error response to DMA	855
28.6.7	Tx DMA configuration	855
28.6.8	Rx DMA configuration	866
28.6.9	DMA interrupts	878
28.7	Ethernet interrupts	879
28.8	Ethernet register descriptions	880
28.8.1	MAC register description	880
28.8.2	MMC register description	899
28.8.3	IEEE 1588 time stamp registers	904
28.8.4	DMA register description	911
28.8.5	Ethernet register maps	925
29	USB on-the-go full-speed (OTG_FS)	929
29.1	OTG_FS introduction	929
29.2	OTG_FS main features	930
29.2.1	General features	930
29.2.2	Host-mode features	931
29.2.3	Peripheral-mode features	931
29.3	OTG_FS functional description	932
29.3.1	OTG full-speed core	932

29.3.2	Full-speed OTG PHY	933
29.4	OTG dual role device (DRD)	934
29.4.1	ID line detection	934
29.4.2	HNP dual role device	934
29.4.3	SRP dual role device	935
29.5	USB peripheral	935
29.5.1	SRP-capable peripheral	936
29.5.2	Peripheral states	936
29.5.3	Peripheral endpoints	937
29.6	USB host	939
29.6.1	SRP-capable host	940
29.6.2	USB host states	940
29.6.3	Host channels	942
29.6.4	Host scheduler	943
29.7	SOF trigger	944
29.7.1	Host SOFs	944
29.7.2	Peripheral SOFs	944
29.8	Power options	945
29.9	Dynamic update of the OTG_FS_HFIR register	946
29.10	USB data FIFOs	946
29.11	Peripheral FIFO architecture	947
29.11.1	Peripheral Rx FIFO	947
29.11.2	Peripheral Tx FIFOs	948
29.12	Host FIFO architecture	948
29.12.1	Host Rx FIFO	948
29.12.2	Host Tx FIFOs	949
29.13	FIFO RAM allocation	949
29.13.1	Device mode	949
29.13.2	Host mode	950
29.14	USB system performance	950
29.15	OTG_FS interrupts	951
29.16	OTG_FS control and status registers	953
29.16.1	CSR memory map	954
29.16.2	OTG_FS global registers	958
29.16.3	Host-mode registers	980

29.16.4	Device-mode registers	991
29.16.5	OTG_FS power and clock gating control register (OTG_FS_PCGCCTL)	1013
29.16.6	OTG_FS register map	1014
29.17	OTG_FS programming model	1020
29.17.1	Core initialization	1020
29.17.2	Host initialization	1022
29.17.3	Device initialization	1022
29.17.4	Host programming model	1023
29.17.5	Device programming model	1040
29.17.6	Operational model	1042
29.17.7	Worst case response time	1059
29.17.8	OTG programming model	1060
30	USB on-the-go high-speed (OTG_HS)	1067
30.1	OTG_HS introduction	1067
30.2	OTG_HS main features	1068
30.2.1	General features	1068
30.2.2	Host-mode features	1069
30.2.3	Peripheral-mode features	1069
30.3	OTG_HS functional description	1070
30.3.1	High-speed OTG PHY	1070
30.3.2	External Full-speed OTG PHY using the I2C interface	1070
30.3.3	Embedded Full-speed OTG PHY	1070
30.4	OTG dual-role device	1071
30.4.1	ID line detection	1071
30.4.2	HNP dual role device	1071
30.4.3	SRP dual-role device	1071
30.5	USB functional description in peripheral mode	1072
30.5.1	SRP-capable peripheral	1072
30.5.2	Peripheral states	1072
30.5.3	Peripheral endpoints	1073
30.6	USB functional description on host mode	1076
30.6.1	SRP-capable host	1076
30.6.2	USB host states	1076
30.6.3	Host channels	1078
30.6.4	Host scheduler	1079

30.7	SOF trigger	1080
30.7.1	Host SOFs	1080
30.7.2	Peripheral SOFs	1080
30.8	USB_HS power modes	1081
30.9	Dynamic update of the OTG_HS_HFIR register	1082
30.10	FIFO RAM allocation	1082
30.10.1	Peripheral mode	1082
30.10.2	Host mode	1083
30.11	OTG_HS interrupts	1083
30.12	OTG_HS control and status registers	1085
30.12.1	CSR memory map	1086
30.12.2	OTG_HS global registers	1091
30.12.3	Host-mode registers	1115
30.12.4	Device-mode registers	1127
30.12.5	OTG_HS power and clock gating control register (OTG_HS_PCGCCTL)	1154
30.12.6	OTG_HS register map	1155
30.13	OTG_HS programming model	1167
30.13.1	Core initialization	1167
30.13.2	Host initialization	1168
30.13.3	Device initialization	1169
30.13.4	DMA mode	1169
30.13.5	Host programming model	1169
30.13.6	Device programming model	1197
30.13.7	Operational model	1199
30.13.8	Worst case response time	1216
30.13.9	OTG programming model	1217
31	Flexible static memory controller (FSMC)	1224
31.1	FSMC main features	1224
31.2	Block diagram	1225
31.3	AHB interface	1225
31.3.1	Supported memories and transactions	1226
31.4	External device address mapping	1227
31.4.1	NOR/PSRAM address mapping	1227
31.4.2	NAND/PC Card address mapping	1228

31.5	NOR Flash/PSRAM controller	1229
31.5.1	External memory interface signals	1230
31.5.2	Supported memories and transactions	1232
31.5.3	General timing rules	1233
31.5.4	NOR Flash/PSRAM controller asynchronous transactions	1233
31.5.5	Synchronous burst transactions	1250
31.5.6	NOR/PSRAM controller registers	1256
31.6	NAND Flash/PC Card controller	1261
31.6.1	External memory interface signals	1262
31.6.2	NAND Flash / PC Card supported memories and transactions	1264
31.6.3	Timing diagrams for NAND and PC Card	1264
31.6.4	NAND Flash operations	1265
31.6.5	NAND Flash pre-wait functionality	1266
31.6.6	Error correction code computation ECC (NAND Flash)	1267
31.6.7	PC Card/CompactFlash operations	1267
31.6.8	NAND Flash/PC Card controller registers	1270
31.6.9	FSMC register map	1276
32	Debug support (DBG)	1278
32.1	Overview	1278
32.2	Reference ARM documentation	1279
32.3	SWJ debug port (serial wire and JTAG)	1279
32.3.1	Mechanism to select the JTAG-DP or the SW-DP	1280
32.4	Pinout and debug port pins	1280
32.4.1	SWJ debug port pins	1281
32.4.2	Flexible SWJ-DP pin assignment	1281
32.4.3	Internal pull-up and pull-down on JTAG pins	1282
32.4.4	Using serial wire and releasing the unused debug pins as GPIOs	1283
32.5	STM32F40x and STM32F41x JTAG TAP connection	1283
32.6	ID codes and locking mechanism	1284
32.6.1	MCU device ID code	1284
32.6.2	Boundary scan TAP	1285
32.6.3	Cortex TM -M4F TAP	1285
32.6.4	Cortex TM -M4F JEDEC-106 ID code	1285
32.7	JTAG debug port	1285
32.8	SW debug port	1287

32.8.1	SW protocol introduction	1287
32.8.2	SW protocol sequence	1287
32.8.3	SW-DP state machine (reset, idle states, ID code)	1288
32.8.4	DP and AP read/write accesses	1289
32.8.5	SW-DP registers	1289
32.8.6	SW-AP registers	1290
32.9	AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP	1290
32.10	Core debug	1291
32.11	Capability of the debugger host to connect under system reset	1292
32.12	FPB (Flash patch breakpoint)	1292
32.13	DWT (data watchpoint trigger)	1293
32.14	ITM (instrumentation trace macrocell)	1293
32.14.1	General description	1293
32.14.2	Time stamp packets, synchronization and overflow packets	1293
32.15	ETM (Embedded trace macrocell)	1295
32.15.1	General description	1295
32.15.2	Signal protocol, packet types	1295
32.15.3	Main ETM registers	1296
32.15.4	Configuration example	1296
32.16	MCU debug component (DBGMCU)	1296
32.16.1	Debug support for low-power modes	1296
32.16.2	Debug support for timers, watchdog, bxCAN and I ² C	1297
32.16.3	Debug MCU configuration register	1297
32.16.4	Debug MCU APB1 freeze register (DBGMCU_APB1_FZ)	1299
32.16.5	Debug MCU APB2 Freeze register (DBGMCU_APB2_FZ)	1300
32.17	TPIU (trace port interface unit)	1301
32.17.1	Introduction	1301
32.17.2	TRACE pin assignment	1301
32.17.3	TPUI formatter	1303
32.17.4	TPUI frame synchronization packets	1304
32.17.5	Transmission of the synchronization frame packet	1304
32.17.6	Synchronous mode	1304
32.17.7	Asynchronous mode	1305
32.17.8	TRACECLKIN connection inside the STM32F40x and STM32F41x .	1305
32.17.9	TPIU registers	1306

32.17.10	Example of configuration	1307
32.18	DBG register map	1307
33	Device electronic signature	1308
33.1	Unique device ID register (96 bits)	1308
33.2	Flash size	1309
	Revision history	1311

List of tables

Table 1.	STM32F40x and STM32F41x register boundary addresses	50
Table 2.	Flash module organization	54
Table 3.	Number of wait states according to CPU clock (HCLK) frequency	55
Table 4.	Boot modes.	58
Table 5.	Memory mapping vs. Boot mode/physical remap.	58
Table 6.	CRC calculation unit register map and reset values.	62
Table 7.	Low-power mode summary	70
Table 8.	Sleep-now.	72
Table 9.	Sleep-on-exit.	72
Table 10.	Stop mode	73
Table 11.	Standby mode.	74
Table 12.	PWR - register map and reset values.	81
Table 13.	RCC register map and reset values	134
Table 14.	Port bit configuration table	137
Table 15.	Flexible SWJ-DP pin assignment	139
Table 16.	RTC_AF1 pin	147
Table 17.	RTC_AF2 pin	147
Table 18.	GPIO register map and reset values	153
Table 19.	SYSCFG register map and reset values.	159
Table 20.	DMA1 request mapping	164
Table 21.	DMA2 request mapping	165
Table 22.	Source and destination address	166
Table 23.	Source and destination address registers in Double buffer mode (DBM=1).	171
Table 24.	Packing/unpacking & endian behavior (bit PINC = MINC = 1)	172
Table 25.	Restriction on NDT versus PSIZE and MSIZE	172
Table 26.	FIFO threshold configurations	174
Table 27.	Possible DMA configurations	178
Table 28.	DMA interrupt requests.	181
Table 29.	DMA register map and reset values	191
Table 30.	Vector table.	195
Table 31.	External interrupt/event controller register map and reset values.	206
Table 32.	ADC pins.	209
Table 33.	Analog watchdog channel selection	212
Table 34.	Configuring the trigger polarity	216
Table 35.	External trigger for regular channels.	216
Table 36.	External trigger for injected channels	217
Table 37.	ADC interrupts	231
Table 38.	ADC global register map.	247
Table 39.	ADC register map and reset values for each ADC	248
Table 40.	ADC register map and reset values (common ADC registers)	249
Table 41.	DAC pins.	251
Table 42.	External triggers	254
Table 43.	DAC register map	269
Table 44.	DCMI pins	270
Table 45.	DCMI signals	272
Table 46.	Positioning of captured data bytes in 32-bit words (8-bit width)	273
Table 47.	Positioning of captured data bytes in 32-bit words (10-bit width)	273
Table 48.	Positioning of captured data bytes in 32-bit words (12-bit width)	273

Table 49.	Positioning of captured data bytes in 32-bit words (14-bit width)	274
Table 50.	Data storage in monochrome progressive video format.	279
Table 51.	Data storage in RGB progressive video format	280
Table 52.	Data storage in YCbCr progressive video format	280
Table 53.	DCMI interrupts.	280
Table 54.	DCMI register map and reset values	291
Table 55.	Counting direction versus encoder signals	325
Table 56.	TIMx Internal trigger connection	339
Table 57.	Output control bits for complementary OC _x and OC _{xN} channels with break feature.	351
Table 58.	TIM1&TIM8 register map and reset values.	359
Table 59.	Counting direction versus encoder signals	385
Table 60.	TIMx internal trigger connection	395
Table 61.	Output control bit for standard OC _x channels.	405
Table 62.	TIMx internal trigger connection	437
Table 63.	Output control bit for standard OC _x channels.	444
Table 64.	TIM9/12 register map and reset values	445
Table 65.	Output control bit for standard OC _x channels.	452
Table 66.	TIM10/11/13/14 register map and reset values	455
Table 67.	TIM6&TIM7 register map and reset values.	466
Table 68.	Min/max IWDG timeout period at 32 kHz (LSI)	468
Table 69.	IWDG register map and reset values	471
Table 70.	Timeout values at 30 MHz (f _{PCLK1})	475
Table 71.	WWDG register map and reset values	478
Table 72.	Data types.	492
Table 73.	CRYP register map and reset values	511
Table 74.	RNG register map and reset map	517
Table 75.	HASH register map and reset values	535
Table 76.	Effect of low power modes on RTC	551
Table 77.	Interrupt control bits	552
Table 78.	RTC register map and reset values	573
Table 79.	SMBus vs. I ₂ C	587
Table 80.	I ₂ C Interrupt requests	591
Table 81.	I ₂ C register map and reset values	605
Table 82.	Noise detection from sampled data	617
Table 83.	Error calculation for programmed baud rates at f _{PCLK} = 8 MHz or f _{PCLK} = 12 MHz, oversampling by 16.	620
Table 84.	Error calculation for programmed baud rates at f _{PCLK} = 8 MHz or f _{PCLK} = 12 MHz, oversampling by 8.	621
Table 85.	Error calculation for programmed baud rates at f _{PCLK} = 16 MHz or f _{PCLK} = 24 MHz, oversampling by 16.	622
Table 86.	Error calculation for programmed baud rates at f _{PCLK} = 16 MHz or f _{PCLK} = 24 MHz, oversampling by 8.	623
Table 87.	Error calculation for programmed baud rates at f _{PCLK} = 8 MHz or f _{PCLK} = 16 MHz, oversampling by 16.	623
Table 88.	Error calculation for programmed baud rates at f _{PCLK} = 8 MHz or f _{PCLK} = 16 MHz, oversampling by 8.	624
Table 89.	Error calculation for programmed baud rates at f _{PCLK} = 30 MHz or f _{PCLK} = 60 MHz, oversampling by 16.	625
Table 90.	Error calculation for programmed baud rates at f _{PCLK} = 30 MHz or f _{PCLK} = 60 MHz, oversampling by 8	625
Table 91.	Error calculation for programmed baud rates at f _{PCLK} = 42 MHz or f _{PCLK} = 84 Hz,	

oversampling by 16	626
Table 92. Error calculation for programmed baud rates at fPCLK = 42 MHz or fPCLK = 84 MHz, oversampling by 8	627
Table 93. USART receiver's tolerance when DIV fraction is 0	628
Table 94. USART receiver's tolerance when DIV_Fraction is different from 0	629
Table 95. Frame formats	631
Table 96. USART interrupt requests	645
Table 97. USART mode configuration	646
Table 98. USART register map and reset values	657
Table 99. SPI interrupt requests	683
Table 100. Audio frequency precision (for PLLM VCO = 1 MHz or 2 MHz)	694
Table 101. I ² S interrupt requests	700
Table 102. SPI register map and reset values	709
Table 103. SDIO I/O definitions	714
Table 104. Command format	718
Table 105. Short response format	719
Table 106. Long response format	719
Table 107. Command path status flags	719
Table 108. Data token format	722
Table 109. Transmit FIFO status flags	723
Table 110. Receive FIFO status flags	724
Table 111. Card status	734
Table 112. SD status	736
Table 113. Speed class code field	738
Table 114. Performance move field	738
Table 115. AU_SIZE field	738
Table 116. Maximum AU size	739
Table 117. Erase size field	739
Table 118. Erase timeout field	739
Table 119. Erase offset field	740
Table 120. Block-oriented write commands	742
Table 121. Block-oriented write protection commands	743
Table 122. Erase commands	743
Table 123. I/O mode commands	743
Table 124. Lock card	744
Table 125. Application-specific commands	744
Table 126. R1 response	745
Table 127. R2 response	745
Table 128. R3 response	746
Table 129. R4 response	746
Table 130. R4b response	746
Table 131. R5 response	747
Table 132. R6 response	748
Table 133. Response type and SDIO_RESPx registers	754
Table 134. SDIO register map	764
Table 135. Transmit mailbox mapping	781
Table 136. Receive mailbox mapping	781
Table 137. bxCAN register map and reset values	808
Table 138. Alternate function mapping	814
Table 139. Management frame format	816
Table 140. Clock range	818
Table 141. TX interface signal encoding	819

Table 142. RX interface signal encoding	819
Table 143. Frame statuses	835
Table 144. Destination address filtering table	841
Table 145. Source address filtering table	842
Table 146. Receive descriptor 0 - encoding for bits 7, 5 and 0 (normal descriptor format only, EDFE=0)	871
Table 148. Ethernet register map and reset values	925
Table 149. Core global control and status registers (CSRs)	954
Table 150. Host-mode control and status registers (CSRs)	955
Table 151. Device-mode control and status registers	956
Table 152. Data FIFO (DFIFO) access register map	958
Table 153. Power and clock gating control and status registers	958
Table 154. Minimum duration for soft disconnect	993
Table 155. OTG_FS register map and reset values	1014
Table 156. Core global control and status registers (CSRs)	1086
Table 157. Host-mode control and status registers (CSRs)	1087
Table 158. Device-mode control and status registers	1088
Table 159. Data FIFO (DFIFO) access register map	1090
Table 160. Power and clock gating control and status registers	1090
Table 161. Minimum duration for soft disconnect	1130
Table 162. OTG_HS register map and reset values	1155
Table 163. NOR/PSRAM bank selection	1227
Table 164. External memory address	1228
Table 165. Memory mapping and timing registers	1228
Table 166. NAND bank selections	1229
Table 167. Programmable NOR/PSRAM access parameters	1230
Table 168. Nonmultipled I/O NOR Flash	1230
Table 169. Multiplexed I/O NOR Flash	1231
Table 170. Nonmultiplexed I/Os PSRAM/SRAM	1231
Table 171. Multiplexed I/O PSRAM	1232
Table 172. NOR Flash/PSRAM supported memories and transactions	1232
Table 173. FSMC_BCRx bit fields	1235
Table 174. FSMC_BTRx bit fields	1235
Table 175. FSMC_BCRx bit fields	1237
Table 176. FSMC_BTRx bit fields	1237
Table 177. FSMC_BWTRx bit fields	1237
Table 178. FSMC_BCRx bit fields	1240
Table 179. FSMC_BTRx bit fields	1240
Table 180. FSMC_BWTRx bit fields	1240
Table 181. FSMC_BCRx bit fields	1242
Table 182. FSMC_BTRx bit fields	1243
Table 183. FSMC_BWTRx bit fields	1243
Table 184. FSMC_BCRx bit fields	1245
Table 185. FSMC_BTRx bit fields	1245
Table 186. FSMC_BWTRx bit fields	1245
Table 187. FSMC_BCRx bit fields	1247
Table 188. FSMC_BTRx bit fields	1247
Table 189. FSMC_BCRx bit fields	1252
Table 190. FSMC_BTRx bit fields	1253
Table 191. FSMC_BCRx bit fields	1255
Table 192. FSMC_BTRx bit fields	1255
Table 193. Programmable NAND/PC Card access parameters	1262

Table 194. 8-bit NAND Flash	1262
Table 195. 16-bit NAND Flash	1263
Table 196. 16-bit PC Card	1263
Table 197. Supported memories and transactions	1264
Table 198. 16-bit PC-Card signals and access type.	1268
Table 199. ECC result relevant bits	1275
Table 200. FSMC register map.	1276
Table 201. SWJ debug port pins	1281
Table 202. Flexible SWJ-DP pin assignment	1281
Table 203. JTAG debug port data registers	1285
Table 204. 32-bit debug port registers addressed through the shifted value A[3:2]	1286
Table 205. Packet request (8-bits)	1287
Table 206. ACK response (3 bits).	1288
Table 207. DATA transfer (33 bits)	1288
Table 208. SW-DP registers	1289
Table 209. Cortex™-M4F AHB-AP registers	1291
Table 210. Core debug registers	1291
Table 211. Main ITM registers	1294
Table 212. Main ETM registers.	1296
Table 213. Asynchronous TRACE pin assignment.	1302
Table 214. Synchronous TRACE pin assignment	1302
Table 215. Flexible TRACE pin assignment.	1303
Table 216. Important TPIU registers.	1306
Table 217. DBG register map and reset values	1307
Table 218. Document revision history	1311

List of figures

Figure 1.	System architecture	48
Figure 2.	CRC calculation unit block diagram	60
Figure 3.	Power supply overview	64
Figure 4.	Backup domain	67
Figure 5.	Power-on reset/power-down reset waveform	68
Figure 6.	BOR thresholds	69
Figure 7.	PVD thresholds	69
Figure 8.	Simplified diagram of the reset circuit	83
Figure 9.	Clock tree	85
Figure 10.	HSE/ LSE clock sources	87
Figure 11.	Frequency measurement with TIM5 in Input capture mode	92
Figure 12.	Frequency measurement with TIM11 in Input capture mode	92
Figure 13.	Basic structure of a five-volt tolerant I/O port bit	137
Figure 14.	Selecting an alternate function	141
Figure 15.	Input floating/pull up/pull down configurations	143
Figure 16.	Output configuration	144
Figure 17.	Alternate function configuration	145
Figure 18.	High impedance-analog configuration	145
Figure 19.	DMA block diagram	162
Figure 20.	System implementation of two DMA controllers	163
Figure 21.	Channel selection	164
Figure 22.	Peripheral-to-memory mode	167
Figure 23.	Memory-to-peripheral mode	168
Figure 24.	Memory-to-memory mode	169
Figure 25.	FIFO structure	174
Figure 26.	External interrupt/event controller block diagram	200
Figure 27.	External interrupt/event GPIO mapping	202
Figure 28.	Single ADC block diagram	208
Figure 29.	Timing diagram	211
Figure 30.	Analog watchdog's guarded area	211
Figure 31.	Injected conversion latency	213
Figure 32.	Right alignment of 12-bit data	215
Figure 33.	Left alignment of 12-bit data	215
Figure 34.	Left alignment of 6-bit data	215
Figure 35.	Multi ADC block diagram ⁽¹⁾	220
Figure 36.	Injected simultaneous mode on 4 channels: dual ADC mode	223
Figure 37.	Injected simultaneous mode on 4 channels: triple ADC mode	223
Figure 38.	Regular simultaneous mode on 16 channels: dual ADC mode	224
Figure 39.	Regular simultaneous mode on 16 channels: triple ADC mode	224
Figure 40.	Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode	225
Figure 41.	Interleaved mode on 1 channel in continuous conversion mode: triple ADC mode	226
Figure 42.	Alternate trigger: injected group of each ADC	227
Figure 43.	Alternate trigger: 4 injected channels (each ADC) in discontinuous mode	227
Figure 44.	Alternate trigger: injected group of each ADC	228
Figure 45.	Alternate + regular simultaneous	229
Figure 46.	Case of trigger occurring during injected conversion	229
Figure 47.	Temperature sensor and VREFINT channel block diagram	230
Figure 48.	DAC channel block diagram	251

Figure 49.	Data registers in single DAC channel mode	252
Figure 50.	Data registers in dual DAC channel mode	253
Figure 51.	Timing diagram for conversion with trigger disabled TEN = 0	253
Figure 52.	DAC LFSR register calculation algorithm	255
Figure 53.	DAC conversion (SW trigger enabled) with LFSR wave generation	256
Figure 54.	DAC triangle wave generation	256
Figure 55.	DAC conversion (SW trigger enabled) with triangle wave generation	257
Figure 56.	DCMI block diagram	271
Figure 57.	Top-level block diagram	271
Figure 58.	DCMI signal waveforms	272
Figure 59.	Timing diagram	274
Figure 60.	Frame capture waveforms in Snapshot mode	276
Figure 61.	Frame capture waveforms in continuous grab mode	277
Figure 62.	Coordinates and size of the window after cropping	277
Figure 63.	Data capture waveforms	278
Figure 64.	Pixel raster scan order	279
Figure 65.	Advanced-control timer block diagram	294
Figure 66.	Counter timing diagram with prescaler division change from 1 to 2	296
Figure 67.	Counter timing diagram with prescaler division change from 1 to 4	296
Figure 68.	Counter timing diagram, internal clock divided by 1	297
Figure 69.	Counter timing diagram, internal clock divided by 2	297
Figure 70.	Counter timing diagram, internal clock divided by 4	298
Figure 71.	Counter timing diagram, internal clock divided by N	298
Figure 72.	Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)	298
Figure 73.	Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)	299
Figure 74.	Counter timing diagram, internal clock divided by 1	300
Figure 75.	Counter timing diagram, internal clock divided by 2	300
Figure 76.	Counter timing diagram, internal clock divided by 4	300
Figure 77.	Counter timing diagram, internal clock divided by N	301
Figure 78.	Counter timing diagram, update event when repetition counter is not used	301
Figure 79.	Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6	302
Figure 80.	Counter timing diagram, internal clock divided by 2	303
Figure 81.	Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36	303
Figure 82.	Counter timing diagram, internal clock divided by N	303
Figure 83.	Counter timing diagram, update event with ARPE=1 (counter underflow)	304
Figure 84.	Counter timing diagram, Update event with ARPE=1 (counter overflow)	304
Figure 85.	Update rate examples depending on mode and TIMx_RCR register settings	305
Figure 86.	Control circuit in normal mode, internal clock divided by 1	306
Figure 87.	TI2 external clock connection example	306
Figure 88.	Control circuit in external clock mode 1	307
Figure 89.	External trigger input block	308
Figure 90.	Control circuit in external clock mode 2	308
Figure 91.	Capture/compare channel (example: channel 1 input stage)	309
Figure 92.	Capture/compare channel 1 main circuit	309
Figure 93.	Output stage of capture/compare channel (channel 1 to 3)	310
Figure 94.	Output stage of capture/compare channel (channel 4)	310
Figure 95.	PWM input mode timing	312
Figure 96.	Output compare mode, toggle on OC1	314
Figure 97.	Edge-aligned PWM waveforms (ARR=8)	315
Figure 98.	Center-aligned PWM waveforms (ARR=8)	316

Figure 99. Complementary output with dead-time insertion	317
Figure 100. Dead-time waveforms with delay greater than the negative pulse.	317
Figure 101. Dead-time waveforms with delay greater than the positive pulse.	318
Figure 102. Output behavior in response to a break.	320
Figure 103. Clearing TIMx OCxREF	321
Figure 104. 6-step generation, COM example (OSSR=1)	322
Figure 105. Example of one pulse mode.	323
Figure 106. Example of counter operation in encoder interface mode.	326
Figure 107. Example of encoder interface mode with TI1FP1 polarity inverted.	326
Figure 108. Example of hall sensor interface.	328
Figure 109. Control circuit in reset mode.	329
Figure 110. Control circuit in gated mode	330
Figure 111. Control circuit in trigger mode.	331
Figure 112. Control circuit in external clock mode 2 + trigger mode	332
Figure 113. Counter timing diagram with prescaler division change from 1 to 2	363
Figure 114. Counter timing diagram with prescaler division change from 1 to 4	363
Figure 115. Counter timing diagram, internal clock divided by 1	364
Figure 116. Counter timing diagram, internal clock divided by 2	364
Figure 117. Counter timing diagram, internal clock divided by 4	365
Figure 118. Counter timing diagram, internal clock divided by N.	365
Figure 119. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded).	365
Figure 120. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded).	366
Figure 121. Counter timing diagram, internal clock divided by 1	367
Figure 122. Counter timing diagram, internal clock divided by 2	367
Figure 123. Counter timing diagram, internal clock divided by 4	367
Figure 124. Counter timing diagram, internal clock divided by N.	368
Figure 125. Counter timing diagram, Update event when repetition counter is not used	368
Figure 126. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6	369
Figure 127. Counter timing diagram, internal clock divided by 2	370
Figure 128. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36	370
Figure 129. Counter timing diagram, internal clock divided by N.	370
Figure 130. Counter timing diagram, Update event with ARPE=1 (counter underflow).	371
Figure 131. Counter timing diagram, Update event with ARPE=1 (counter overflow).	371
Figure 132. Control circuit in normal mode, internal clock divided by 1	372
Figure 133. TI2 external clock connection example.	373
Figure 134. Control circuit in external clock mode 1	373
Figure 135. External trigger input block.	374
Figure 136. Control circuit in external clock mode 2	374
Figure 137. Capture/compare channel (example: channel 1 input stage)	375
Figure 138. Capture/compare channel 1 main circuit	375
Figure 139. Output stage of capture/compare channel (channel 1).	376
Figure 140. PWM input mode timing	378
Figure 141. Output compare mode, toggle on OC1.	380
Figure 142. Edge-aligned PWM waveforms (ARR=8)	381
Figure 143. Center-aligned PWM waveforms (ARR=8)	382
Figure 144. Example of one-pulse mode.	383
Figure 145. Clearing TIMx OCxREF	384
Figure 146. Example of counter operation in encoder interface mode	386
Figure 147. Example of encoder interface mode with TI1FP1 polarity inverted	386
Figure 148. Control circuit in reset mode	388
Figure 149. Control circuit in gated mode	388

Figure 150. Control circuit in trigger mode	389
Figure 151. Control circuit in external clock mode 2 + trigger mode	390
Figure 152. Master/Slave timer example	391
Figure 153. General-purpose timer block diagram (TIM9 and TIM12)	415
Figure 154. General-purpose timer block diagram (TIM10/11/13/14)	416
Figure 155. Counter timing diagram with prescaler division change from 1 to 2	418
Figure 156. Counter timing diagram with prescaler division change from 1 to 4	418
Figure 157. Counter timing diagram, internal clock divided by 1	419
Figure 158. Counter timing diagram, internal clock divided by 2	419
Figure 159. Counter timing diagram, internal clock divided by 4	420
Figure 160. Counter timing diagram, internal clock divided by N	420
Figure 161. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)	420
Figure 162. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)	421
Figure 163. Control circuit in normal mode, internal clock divided by 1	422
Figure 164. TI2 external clock connection example	422
Figure 165. Control circuit in external clock mode 1	423
Figure 166. Capture/compare channel (example: channel 1 input stage)	423
Figure 167. Capture/compare channel 1 main circuit	424
Figure 168. Output stage of capture/compare channel (channel 1)	424
Figure 169. PWM input mode timing	426
Figure 170. Output compare mode, toggle on OC1	428
Figure 171. Edge-aligned PWM waveforms (ARR=8)	429
Figure 172. Example of one pulse mode	429
Figure 173. Control circuit in reset mode	431
Figure 174. Control circuit in gated mode	432
Figure 175. Control circuit in trigger mode	432
Figure 176. Basic timer block diagram	456
Figure 177. Counter timing diagram with prescaler division change from 1 to 2	458
Figure 178. Counter timing diagram with prescaler division change from 1 to 4	458
Figure 179. Counter timing diagram, internal clock divided by 1	459
Figure 180. Counter timing diagram, internal clock divided by 2	459
Figure 181. Counter timing diagram, internal clock divided by 4	460
Figure 182. Counter timing diagram, internal clock divided by N	460
Figure 183. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)	460
Figure 184. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)	461
Figure 185. Control circuit in normal mode, internal clock divided by 1	461
Figure 186. Independent watchdog block diagram	468
Figure 187. Watchdog block diagram	473
Figure 188. Window watchdog timing diagram	474
Figure 189. Block diagram	480
Figure 190. DES/TDES-ECB mode encryption	482
Figure 191. DES/TDES-ECB mode decryption	482
Figure 192. DES/TDES-CBC mode encryption	484
Figure 193. DES/TDES-CBC mode decryption	485
Figure 194. AES-ECB mode encryption	486
Figure 195. AES-ECB mode decryption	487
Figure 196. AES-CBC mode encryption	488
Figure 197. AES-CBC mode decryption	489

Figure 198. AES-CTR mode encryption	490
Figure 199. AES-CTR mode encryption	491
Figure 200. Initial counter block structure for the Counter mode	491
Figure 201. 64-bit block construction according to DATATYPE	494
Figure 202. Initialization vectors use in the TDES-CBC encryption	496
Figure 203. CRYP interrupt mapping diagram	500
Figure 204. Block diagram	513
Figure 205. Block diagram	519
Figure 206. Bit, byte and half-word swapping	521
Figure 207. HASH interrupt mapping diagram	526
Figure 208. RTC block diagram	538
Figure 209. I2C bus protocol	577
Figure 210. I2C block diagram	577
Figure 211. Transfer sequence diagram for slave transmitter	579
Figure 212. Transfer sequence diagram for slave receiver	580
Figure 213. Transfer sequence diagram for master transmitter	582
Figure 214. Transfer sequence diagram for master receiver	584
Figure 215. I2C interrupt mapping diagram	592
Figure 216. USART block diagram	609
Figure 217. Word length programming	610
Figure 218. Configurable stop bits	612
Figure 219. TC/TXE behavior when transmitting	613
Figure 220. Start bit detection when oversampling by 16 or 8	614
Figure 221. Data sampling when oversampling by 16	617
Figure 222. Data sampling when oversampling by 8	617
Figure 223. Mute mode using Idle line detection	630
Figure 224. Mute mode using address mark detection	630
Figure 225. Break detection in LIN mode (11-bit break length - LBDL bit is set)	633
Figure 226. Break detection in LIN mode vs. Framing error detection	634
Figure 227. USART example of synchronous transmission	635
Figure 228. USART data clock timing diagram (M=0)	635
Figure 229. USART data clock timing diagram (M=1)	636
Figure 230. RX data setup/hold time	636
Figure 231. ISO 7816-3 asynchronous protocol	637
Figure 232. Parity error detection using the 1.5 stop bits	638
Figure 233. IrDA SIR ENDEC- block diagram	640
Figure 234. IrDA data modulation (3/16) -Normal mode	640
Figure 235. Transmission using DMA	642
Figure 236. Reception using DMA	643
Figure 237. Hardware flow control between 2 USARTs	643
Figure 238. RTS flow control	644
Figure 239. CTS flow control	644
Figure 240. USART interrupt mapping diagram	645
Figure 241. SPI block diagram	661
Figure 242. Single master/ single slave application	662
Figure 243. Data clock timing diagram	664
Figure 244. TI mode - Slave mode, single transfer	666
Figure 245. TI mode - Slave mode, continuous transfer	666
Figure 246. TI mode - master mode, single transfer	668
Figure 247. TI mode - master mode, continuous transfer	668
Figure 248. TXE/RXNE/BSY behavior in Master / full-duplex mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers	672

Figure 249. TXE/RXNE/BSY behavior in Slave / full-duplex mode (BIDIMODE=0, RXONLY=0) in the case of continuous transfers	672
Figure 250. TXE/BSY behavior in Master transmit-only mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers	673
Figure 251. TXE/BSY in Slave transmit-only mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers	674
Figure 252. RXNE behavior in receive-only mode (BIDIRMODE=0 and RXONLY=1) in the case of continuous transfers	675
Figure 253. TXE/BSY behavior when transmitting (BIDIRMODE=0 and RXONLY=0) in the case of discontinuous transfers	676
Figure 254. Transmission using DMA	681
Figure 255. Reception using DMA	681
Figure 256. TI mode frame format error detection	683
Figure 257. I ² S block diagram	684
Figure 258. I ² S full duplex block diagram	685
Figure 259. I ² S Phillips protocol waveforms (16/32-bit full accuracy, CPOL = 0)	687
Figure 260. I ² S Phillips standard waveforms (24-bit frame with CPOL = 0)	687
Figure 261. Transmitting 0x8EAA33	687
Figure 262. Receiving 0x8EAA33	688
Figure 263. I ² S Phillips standard (16-bit extended to 32-bit packet frame with CPOL = 0)	688
Figure 264. Example	688
Figure 265. MSB Justified 16-bit or 32-bit full-accuracy length with CPOL = 0	689
Figure 266. MSB Justified 24-bit frame length with CPOL = 0	689
Figure 267. MSB Justified 16-bit extended to 32-bit packet frame with CPOL = 0	689
Figure 268. LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0	690
Figure 269. LSB Justified 24-bit frame length with CPOL = 0	690
Figure 270. Operations required to transmit 0x3478AE	690
Figure 271. Operations required to receive 0x3478AE	691
Figure 272. LSB Justified 16-bit extended to 32-bit packet frame with CPOL = 0	691
Figure 273. Example of LSB justified 16-bit extended to 32-bit packet frame	691
Figure 274. PCM standard waveforms (16-bit)	692
Figure 275. PCM standard waveforms (16-bit extended to 32-bit packet frame)	692
Figure 276. Audio sampling frequency definition	693
Figure 277. I ² S clock generator architecture	693
Figure 278. SDIO “no response” and “no data” operations	711
Figure 279. SDIO (multiple) block read operation	711
Figure 280. SDIO (multiple) block write operation	711
Figure 281. SDIO sequential read operation	712
Figure 282. SDIO sequential write operation	712
Figure 283. SDIO block diagram	713
Figure 284. SDIO adapter	714
Figure 285. Control unit	715
Figure 286. SDIO adapter command path	716
Figure 287. Command path state machine (CPSM)	717
Figure 288. SDIO command transfer	718
Figure 289. Data path	720
Figure 290. Data path state machine (DPSM)	721
Figure 291. CAN network topology	767
Figure 292. Dual CAN block diagram	769
Figure 293. bxCAN operating modes	771
Figure 294. bxCAN in silent mode	772
Figure 295. bxCAN in loop back mode	772

Figure 296. bxCAN in combined mode	773
Figure 297. Transmit mailbox states	774
Figure 298. Receive FIFO states	775
Figure 299. Filter bank scale configuration - register organization	778
Figure 300. Example of filter numbering	779
Figure 301. Filtering mechanism - example	780
Figure 302. CAN error state diagram	781
Figure 303. Bit timing	783
Figure 304. CAN frames	784
Figure 305. Event flags and interrupt generation	785
Figure 306. ETH block diagram	815
Figure 307. SMI interface signals	816
Figure 308. MDIO timing and frame structure - Write cycle	817
Figure 309. MDIO timing and frame structure - Read cycle	817
Figure 310. Media independent interface signals	818
Figure 311. MII clock sources	820
Figure 312. Reduced media-independent interface signals	821
Figure 313. RMII clock sources	821
Figure 314. Clock scheme	822
Figure 315. Address field format	824
Figure 316. MAC frame format	825
Figure 317. Tagged MAC frame format	826
Figure 318. Transmission bit order	832
Figure 319. Transmission with no collision	832
Figure 320. Transmission with collision	833
Figure 321. Frame transmission in MMI and RMII modes	833
Figure 322. Receive bit order	837
Figure 323. Reception with no error	838
Figure 324. Reception with errors	838
Figure 325. Reception with false carrier indication	838
Figure 326. MAC core interrupt masking scheme	839
Figure 327. Wakeup frame filter register	844
Figure 328. Networked time synchronization	847
Figure 329. System time update using the Fine correction method	849
Figure 330. PTP trigger output to TIM2 ITR1 connection	851
Figure 331. PPS output	852
Figure 332. Descriptor ring and chain structure	853
Figure 333. TxDMA operation in Default mode	857
Figure 334. TxDMA operation in OSF mode	859
Figure 335. Normal transmit descriptor	860
Figure 336. Enhanced transmit descriptor	865
Figure 337. Receive DMA operation	867
Figure 338. Normal Rx DMA descriptor structure	869
Figure 339. Enhanced receive descriptor field format with IEEE1588 time stamp enabled	876
Figure 340. Interrupt scheme	879
Figure 341. Ethernet MAC remote wakeup frame filter register (ETH_MACRWUFFR)	889
Figure 342. Block diagram	932
Figure 343. OTG A-B device connection	934
Figure 344. USB peripheral-only connection	936
Figure 345. USB host-only connection	940
Figure 346. SOF connectivity	944
Figure 347. Updating OTG_FS_HFIR dynamically	946

Figure 348. Device-mode FIFO address mapping and AHB FIFO access mapping	947
Figure 349. Host-mode FIFO address mapping and AHB FIFO access mapping	948
Figure 350. Interrupt hierarchy.	952
Figure 351. CSR memory map	954
Figure 352. Transmit FIFO write task	1024
Figure 353. Receive FIFO read task	1025
Figure 354. Normal bulk/control OUT/SETUP and bulk/control IN transactions	1027
Figure 355. Bulk/control IN transactions	1030
Figure 356. Normal interrupt OUT/IN transactions	1032
Figure 357. Normal isochronous OUT/IN transactions	1037
Figure 358. Receive FIFO packet read	1043
Figure 359. Processing a SETUP packet	1045
Figure 360. Bulk OUT transaction	1051
Figure 361. TRDT max timing case	1060
Figure 362. A-device SRP	1061
Figure 363. B-device SRP	1062
Figure 364. A-device HNP	1063
Figure 365. B-device HNP	1065
Figure 366. USB OTG interface block diagram	1070
Figure 367. Updating OTG_HS_HFIR dynamically	1082
Figure 368. Interrupt hierarchy.	1084
Figure 369. CSR memory map	1086
Figure 370. Transmit FIFO write task	1172
Figure 371. Receive FIFO read task	1173
Figure 372. Normal bulk/control OUT/SETUP and bulk/control IN transactions - DMA mode	1175
Figure 373. Normal bulk/control OUT/SETUP and bulk/control IN transactions - Slave mode	1176
Figure 374. Bulk/control IN transactions - DMA mode	1179
Figure 375. Bulk/control IN transactions - Slave mode	1180
Figure 376. Normal interrupt OUT/IN transactions - DMA mode	1182
Figure 377. Normal interrupt OUT/IN transactions - Slave mode	1183
Figure 378. Normal isochronous OUT/IN transactions - DMA mode	1188
Figure 379. Normal isochronous OUT/IN transactions - Slave mode	1189
Figure 380. Receive FIFO packet read in slave mode.	1200
Figure 381. Processing a SETUP packet	1202
Figure 382. Slave mode bulk OUT transaction	1208
Figure 383. TRDT max timing case	1217
Figure 384. A-device SRP	1218
Figure 385. B-device SRP	1219
Figure 386. A-device HNP	1220
Figure 387. B-device HNP	1222
Figure 388. FSMC block diagram	1225
Figure 389. FSMC memory banks	1227
Figure 390. Mode1 read accesses.	1234
Figure 391. Mode1 write accesses	1234
Figure 392. ModeA read accesses	1236
Figure 393. ModeA write accesses	1236
Figure 394. Mode2/B read accesses	1238
Figure 395. Mode2 write accesses	1239
Figure 396. ModeB write accesses	1239
Figure 397. ModeC read accesses	1241

Figure 398. ModeC write accesses	1242
Figure 399. ModeD read accesses	1244
Figure 400. ModeD write accesses	1244
Figure 401. Multiplexed read accesses	1246
Figure 402. Multiplexed write accesses	1247
Figure 403. Asynchronous wait during a read access	1249
Figure 404. Asynchronous wait during a write access	1249
Figure 405. Wait configurations	1251
Figure 406. Synchronous multiplexed read mode - NOR, PSRAM (CRAM)	1252
Figure 407. Synchronous multiplexed write mode - PSRAM (CRAM)	1254
Figure 408. NAND/PC Card controller timing for common memory access	1265
Figure 409. Access to non ‘CE don’t care’ NAND-Flash	1266
Figure 410. Block diagram of STM32 MCU and Cortex™-M4F-level debug support	1278
Figure 411. SWJ debug port	1280
Figure 412. JTAG TAP connections	1284
Figure 413. TPIU block diagram	1301

1 Documentation conventions

1.1 List of abbreviations for registers

The following abbreviations are used in register descriptions:

read/write (rw)	Software can read and write to these bits.
read-only (r)	Software can only read these bits.
write-only (w)	Software can only write to this bit. Reading the bit returns the reset value.
read/clear (rc_w1)	Software can read as well as clear this bit by writing 1. Writing '0' has no effect on the bit value.
read/clear (rc_w0)	Software can read as well as clear this bit by writing 0. Writing '1' has no effect on the bit value.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to '0'. Writing '0' has no effect on the bit value.
read/set (rs)	Software can read as well as set this bit. Writing '0' has no effect on the bit value.
read-only write trigger (rt_w)	Software can read this bit. Writing '0' or '1' triggers an event but has no effect on the bit value.
toggle (t)	Software can only toggle this bit by writing '1'. Writing '0' has no effect.
Reserved (Res.)	Reserved bit, must be kept at reset value.

1.2 Peripheral availability

For peripheral availability and number across all STM32F40x and STM32F41x sales types, please refer to the STM32F40x and STM32F41x datasheets.

2 Memory and bus architecture

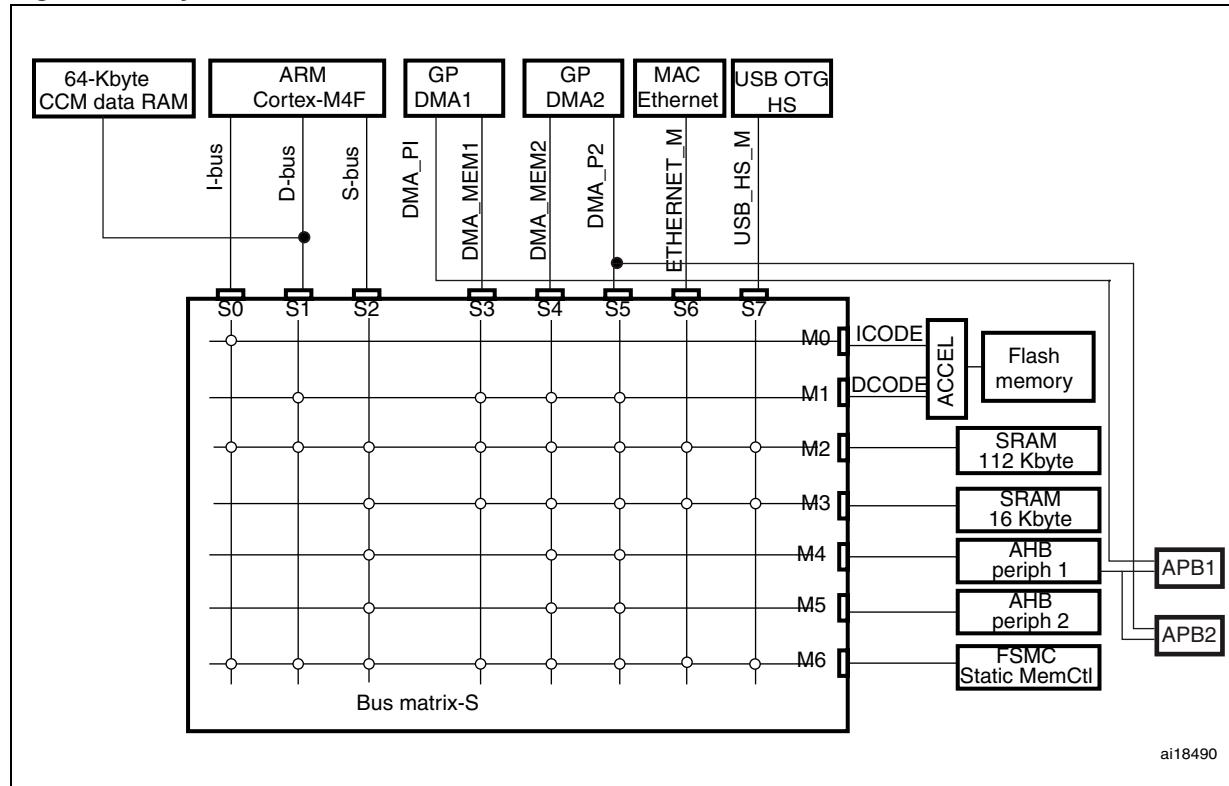
2.1 System architecture

The main system consists of 32-bit multilayer AHB bus matrix that interconnects:

- Height masters:
 - Cortex™-M4F core I-bus, D-bus and S-bus
 - DMA1 memory bus
 - DMA2 memory bus
 - DMA2 peripheral bus
 - Ethernet DMA bus
 - USB OTG HS DMA bus
- Seven slaves:
 - Internal Flash memory ICode bus
 - Internal Flash memory DCode bus
 - Main internal SRAM1 (112 KB)
 - Auxiliary internal SRAM2 (16 KB)
 - AHB1peripherals including AHB to APB bridges and APB peripherals
 - AHB2 peripherals
 - FSMC

The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously. This architecture is shown in [Figure 1](#).

Note: The 64-Kbyte CCM (core coupled memory) data RAM is not part of the bus matrix (see [Figure 1: System architecture](#)). It can be accessed only through the CPU.

Figure 1. System architecture

2.1.1 S0: I-bus

This bus connects the Instruction bus of the Cortex™-M4F core to the BusMatrix. This bus is used by the core to fetch instructions. The target of this bus is a memory containing code (internal Flash memory/SRAM or external memories through the FSMC).

2.1.2 S1: D-bus

This bus connects the databus of the Cortex™-M4F and the 64-Kbyte CCM data RAM to the BusMatrix. This bus is used by the core for literal load and debug access. The target of this bus is a memory containing code or data (internal Flash memory or external memories through the FSMC).

2.1.3 S2: S-bus

This bus connects the system bus of the Cortex™-M4F core to a BusMatrix. This bus is used to access data located in a peripheral or in SRAM. Instructions may also be fetched on this bus (less efficient than ICode). The targets of this bus are the 112 KB and 16 KB internal SRAMs, the AHB1 peripherals including the APB peripherals, the AHB2 peripherals and the external memories through the FSMC.

2.1.4 S3, S4: DMA memory bus

This bus connects the DMA memory bus master interface to the BusMatrix. It is used by the DMA to perform transfer to/from memories. The targets of this bus are data memories: internal SRAM and external memories through the FSMC.

2.1.5 S5: DMA peripheral bus

This bus connects the DMA peripheral master bus interface to the BusMatrix. This bus is used by the DMA to access AHB peripherals or to perform memory-to-memory transfers. The targets of this bus are the AHB and APB peripherals plus data memories: internal SRAM and external memories through the FSMC.

2.1.6 S6: Ethernet DMA bus

This bus connects the Ethernet DMA master interface to the BusMatrix. This bus is used by the Ethernet DMA to load/store data to a memory. The targets of this bus are data memories: internal SRAM and external memories through the FSMC.

2.1.7 S7: USB OTG HS DMA bus

This bus connects the USB OTG HS DMA master interface to the BusMatrix. This bus is used by the USB OTG DMA to load/store data to a memory. The targets of this bus are data memories: internal SRAM and external memories through the FSMC.

2.1.8 BusMatrix

The BusMatrix manages the access arbitration between masters. The arbitration uses a round-robin algorithm.

2.1.9 AHB/APB bridges (APB)

The two AHB/APB bridges, APB1 and APB2, provide full synchronous connections between the AHB and the two APB buses, allowing flexible selection of the peripheral frequency.

Refer to the device datasheets for more details on APB1 and APB2 maximum frequencies, and to [Table 1 on page 50](#) for the address mapping of AHB and APB peripherals.

After each device reset, all peripheral clocks are disabled (except for the SRAM and Flash memory interface). Before using a peripheral you have to enable its clock in the RCC_AHBxENR or RCC_APBxENR register.

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

2.2 Memory organization

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

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

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

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

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

2.3 Memory map

See the datasheet corresponding to your device for a comprehensive diagram of the memory map. [Table 1](#) gives the boundary addresses of the peripherals available in all STM32F40x and STM32F41x devices.

Table 1. STM32F40x and STM32F41x register boundary addresses

Boundary address	Peripheral	Bus	Register map	
0xA000 0000 - 0xA000 0FFF	FSMC control register	AHB3	Section 31.6.9: FSMC register map on page 1276	
0x5006 0800 - 0X5006 0BFF	RNG	AHB2	Section 20.4.4: RNG register map on page 517	
0x5006 0400 - 0X5006 07FF	HASH		Section 21.4.8: HASH register map on page 535	
0x5006 0000 - 0X5006 03FF	CRYP		Section 19.6.11: CRYP register map on page 511	
0x5005 0000 - 0X5005 03FF	DCMI		Section 12.8.12: DCMI register map on page 291	
0x5000 0000 - 0X5003 FFFF	USB OTG FS		Section 29.16.6: OTG_FS register map on page 1014	
0x4004 0000 - 0x4007 FFFF	USB OTG HS	AHB1	Section 30.12.6: OTG_HS register map on page 1155	
0x4002 9000 - 0x4002 93FF	ETHERNET MAC		Section 28.8.5: Ethernet register maps on page 925	
0x4002 8C00 - 0x4002 8FFF				
0x4002 8800 - 0x4002 8BFF				
0x4002 8400 - 0x4002 87FF				
0x4002 8000 - 0x4002 83FF				
0x4002 6400 - 0x4002 67FF	DMA2		Section 8.5.11: DMA register map on page 191	
0x4002 6000 - 0x4002 63FF	DMA1			
0x4002 4000 - 0x4002 4FFF	BKPSRAM			
0x4002 3C00 - 0x4002 3FFF	Flash interface register		See Flash programming manual	
0x4002 3800 - 0x4002 3BFF	RCC		Section 5.3.24: RCC register map on page 134	
0x4002 3000 - 0x4002 33FF	CRC		Section 3.4.4: CRC register map on page 62	
0x4002 2000 - 0x4002 23FF	GPIOI			
0x4002 1C00 - 0x4002 1FFF	GPIOH			
0x4002 1800 - 0x4002 1BFF	GPIOG			
0x4002 1400 - 0x4002 17FF	GPIOF			
0x4002 1000 - 0x4002 13FF	GPIOE			
0X4002 0C00 - 0x4002 0FFF	GPIOD			
0x4002 0800 - 0x4002 0BFF	GPIOC			
0x4002 0400 - 0x4002 07FF	GPIOB			
0x4002 0000 - 0x4002 03FF	GPIOA			

Table 1. STM32F40x and STM32F41x register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4001 4800 - 0x4001 4BFF	TIM11	APB2	Section 15.6.12: TIM10/11/13/14 register map on page 455
0x4001 4400 - 0x4001 47FF	TIM10		Section 15.5.14: TIM9/12 register map on page 445
0x4001 4000 - 0x4001 43FF	TIM9		Section 9.3.7: EXTI register map on page 206
0x4001 3C00 - 0x4001 3FFF	EXTI		Section 7.2.8: SYSCFG register maps on page 160
0x4001 3800 - 0x4001 3BFF	SYSCFG		Section 25.5.10: SPI register map on page 709
0x4001 3000 - 0x4001 33FF	SPI1		Section 26.9.16: SDIO register map on page 764
0x4001 2C00 - 0x4001 2FFF	SDIO		Section 10.13.18: ADC register map on page 247
0x4001 2000 - 0x4001 23FF	ADC1 - ADC2 - ADC3		Section 24.6.8: USART register map on page 657
0x4001 1400 - 0x4001 17FF	USART6		Section 13.4.21: TIM1&TIM8 register map on page 359
0x4001 1000 - 0x4001 13FF	USART1		
0x4001 0400 - 0x4001 07FF	TIM8		
0x4001 0000 - 0x4001 03FF	TIM1		

Table 1. STM32F40x and STM32F41x register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4000 7400 - 0x4000 77FF	DAC	APB1	Section 11.5.15: DAC register map on page 269
0x4000 7000 - 0x4000 73FF	PWR		Section 4.4.3: PWR register map on page 81
0x4000 6800 - 0x4000 6BFF	CAN2		Section 27.9.5: bxCAN register map on page 807
0x4000 6400 - 0x4000 67FF	CAN1		
0x4000 5C00 - 0x4000 5FFF	I2C3		Section 23.6.10: I2C register map on page 605
0x4000 5800 - 0x4000 5BFF	I2C2		
0x4000 5400 - 0x4000 57FF	I2C1		
0x4000 5000 - 0x4000 53FF	UART5		
0x4000 4C00 - 0x4000 4FFF	UART4		
0x4000 4800 - 0x4000 4BFF	USART3		
0x4000 4400 - 0x4000 47FF	USART2		
0x4000 4000 - 0x4000 43FF	I2S3ext		
0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3		Section 25.5.10: SPI register map on page 709
0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2		
0x4000 3400 - 0x4000 37FF	I2S2ext		
0x4000 3000 - 0x4000 33FF	IWDG		Section 17.4.5: IWDG register map on page 471
0x4000 2C00 - 0x4000 2FFF	WWDG		Section 18.6.4: WWWDG register map on page 478
0x4000 2800 - 0x4000 2BFF	RTC & BKP Registers		Section 22.6.21: RTC register map on page 573
0x4000 2000 - 0x4000 23FF	TIM14		Section 15.6.12: TIM10/11/13/14 register map on page 455
0x4000 1C00 - 0x4000 1FFF	TIM13		
0x4000 1800 - 0x4000 1BFF	TIM12		Section 15.5.14: TIM9/12 register map on page 445
0x4000 1400 - 0x4000 17FF	TIM7		Section 16.4.9: TIM6&TIM7 register map on page 466
0x4000 1000 - 0x4000 13FF	TIM6		
0x4000 0C00 - 0x4000 0FFF	TIM5		
0x4000 0800 - 0x4000 0BFF	TIM4		
0x4000 0400 - 0x4000 07FF	TIM3		
0x4000 0000 - 0x4000 03FF	TIM2		Section 14.4.21: TIMx register map on page 412

2.3.1 Embedded SRAM

The STM32F40x and STM32F41x feature 4 Kbytes of backup SRAM (see [Section 4.1.2: Battery backup domain](#)) plus 192 Kbytes of system SRAM.

The system SRAM can be accessed as bytes, half-words (16 bits) or full words (32 bits). The start address of the SRAM is 0x2000 0000. Read and write operations are performed at CPU speed with 0 wait state.

The system SRAM is split up into three blocks, of 112 KB, 64 KB, and 16 KB, with a capability for concurrent access from by the AHB masters (like the Ethernet or the USB OTG

HS): for instance, the Ethernet MAC can read/write from/to the 16 KB SRAM while the CPU is reading/writing from/to the 112 KB SRAM.

The CPU can access the system SRAM through the System Bus or through the I-Code/D-Code buses when boot from SRAM is selected or when physical remap is selected ([Section 7.2.1: SYSCFG memory remap register \(SYSCFG_MEMRMP\)](#) in the SYSCFG controller). To get the max performance on SRAM execution, physical remap should be selected (boot or software selection).

2.3.2 Bit banding

The Cortex™-M4F memory map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

In the STM32F40x and STM32F41x both the peripheral registers and the SRAM are mapped to a bit-band region, so that single bit-band write and read operations are allowed. The operations are only available for Cortex™-M4F accesses, and not from other bus masters (e.g. DMA).

A mapping formula shows how to reference each word in the alias region to a corresponding bit in the bit-band region. The mapping formula is:

$$\text{bit_word_addr} = \text{bit_band_base} + (\text{byte_offset} \times 32) + (\text{bit_number} \times 4)$$

where:

- *bit_word_addr* is the address of the word in the alias memory region that maps to the targeted bit
- *bit_band_base* is the starting address of the alias region
- *byte_offset* is the number of the byte in the bit-band region that contains the targeted bit
- *bit_number* is the bit position (0-7) of the targeted bit

Example

The following example shows how to map bit 2 of the byte located at SRAM address 0x20000300 to the alias region:

$$0x22006008 = 0x22000000 + (0x300*32) + (2*4)$$

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

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

For more information on bit-banding, please refer to the *Cortex™-M4F programming manual* (see [Related documents on page 1](#)).

2.3.3 Embedded Flash memory

The Flash memory has the following main features:

- Capacity up to 1 Mbyte
- 128 bits wide data read
- Byte, half-word, word and double word write
- Sector and mass erase
- Memory organization

The Flash memory is organized as follows:

- A main memory block divided into 4 sectors of 16 Kbytes, 1 sector of 64 Kbytes, and 7 sectors of 128 Kbytes
- System memory from which the device boots in System memory boot mode
- 512 OTP (one-time programmable) bytes for user data
The OTP area contains 16 additional bytes used to lock the corresponding OTP data block.
- Option bytes to configure read and write protection, BOR level, watchdog software/hardware and reset when the device is in Standby or Stop mode

Table 2. Flash module organization

Block	Name	Block base addresses	Size
Main memory	Sector 0	0x0800 0000 - 0x0800 3FFF	16 Kbyte
	Sector 1	0x0800 4000 - 0x0800 7FFF	16 Kbyte
	Sector 2	0x0800 8000 - 0x0800 BFFF	16 Kbyte
	Sector 3	0x0800 C000 - 0x0800 FFFF	16 Kbyte
	Sector 4	0x0801 0000 - 0x0801 FFFF	64 Kbyte
	Sector 5	0x0802 0000 - 0x0803 FFFF	128 Kbyte
	Sector 6	0x0804 0000 - 0x0805 FFFF	128 Kbyte
	.	.	.
	.	.	.
	.	.	.
	Sector 11	0x080E 0000 - 0x080F FFFF	128 Kbyte
	System memory	0x1FFF 0000 - 0x1FFF 77FF	30 Kbyte
OTP	OTP	0x1FFF 7800 - 0x1FFF 7A0F	528 bytes
	Option bytes	0x1FFF C000 - 0x1FFF C00F	16 bytes

2.3.4 Flash memory read interface

Relation between CPU clock frequency and Flash memory read time

To correctly read data from Flash memory, the number of wait states (LATENCY) must be correctly programmed in the *Flash access control register (FLASH_ACR)* according to the frequency of the Cortex™-M4F clock and the supply voltage of the device. *Table 3* shows

the correspondence between wait states and core clock frequency.

Note: When $V_{OS} = '0'$, the maximum value of $f_{HCLK} = 144 \text{ MHz}$.

Table 3. Number of wait states according to CPU clock (HCLK) frequency

Wait states (WS) (LATENCY)	HCLK (MHz)			
	Voltage range 2.7 V - 3.6 V	Voltage range 2.4 V - 2.7 V	Voltage range 2.1 V - 2.4 V	Voltage range 1.8 V - 2.1 V ⁽¹⁾
0 WS (1 CPU cycle)	$0 < HCLK \leq 30$	$0 < HCLK \leq 24$	$0 < HCLK \leq 18$	$0 < HCLK \leq 16$
1 WS (2 CPU cycles)	$30 < HCLK \leq 60$	$24 < HCLK \leq 48$	$18 < HCLK \leq 36$	$16 < HCLK \leq 32$
2 WS (3 CPU cycles)	$60 < HCLK \leq 90$	$48 < HCLK \leq 72$	$36 < HCLK \leq 54$	$32 < HCLK \leq 48$
3 WS (4 CPU cycles)	$90 < HCLK \leq 120$	$72 < HCLK \leq 96$	$54 < HCLK \leq 72$	$48 < HCLK \leq 64$
4 WS (5 CPU cycles)	$120 < HCLK \leq 150$	$96 < HCLK \leq 120$	$72 < HCLK \leq 90$	$64 < HCLK \leq 80$
5 WS (6 CPU cycles)	$150 < HCLK \leq 168$	$120 < HCLK \leq 144$	$90 < HCLK \leq 108$	$80 < HCLK \leq 96$
6 WS (7 CPU cycles)		$144 < HCLK \leq 168$	$108 < HCLK \leq 120$	$96 < HCLK \leq 112$
7 WS (8 CPU cycles)			$120 < HCLK \leq 138$	$112 < HCLK \leq 128$

1. If PDR_ON is set to V_{SS} , this value can be lowered to 1.7 V when the device operates in the 0 to 70 °C.

After reset, the CPU clock frequency is 16 MHz and 0 wait state (WS) is configured in the FLASH_ACR register.

It is highly recommended to use the following software sequences to tune the number of wait states needed to access the Flash memory with the CPU frequency.

Increasing the CPU frequency

- Program the new number of wait states to the LATENCY bits in the FLASH_ACR register
- Check that the new number of wait states is used to access the Flash memory by reading the FLASH_ACR register
- Modify the CPU clock source by writing the SW bits in the *RCC clock configuration register (RCC_CFGR)*
- If needed, modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR
- Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register

Decreasing the CPU frequency

- Modify the CPU clock source by writing the SW bits in the RCC_CFGR register
- If needed, modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR
- Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register
- Program the new number of wait states to the LATENCY bits in FLASH_ACR
- Check that the new number of wait states is used to access the Flash memory by reading the FLASH_ACR register

Note:

A change in CPU clock configuration or wait state (WS) configuration may not be effective straight away. To make sure that the current CPU clock frequency is the one you have configured, you can check the AHB prescaler factor and clock source status values. To make sure that the number of WS you have programmed is effective, you can read the FLASH_ACR register.

The FLASH_ACR register is used to enable/disable the acceleration features and control the Flash memory access time according to CPU frequency. The tables below provides the bit map and bit descriptions for this register.

For complete information on Flash memory operations and register configurations, please refer to the STM32F40x and STM32F41x Flash programming manual (PM0059).

Flash access control register (FLASH_ACR)

The Flash access control register is used to enable/disable the acceleration features and control the Flash memory access time according to CPU frequency.

Address offset: 0x00

Reset value: 0x0000 0000

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

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

Bits 31:11 Reserved, must be kept cleared.

Bit 12 **DCRST:** Data cache reset

- 0: Data cache is not reset
- 1: Data cache is reset

This bit can be written only when the D cache is disabled.

Bit 11 **ICRST:** Instruction cache reset

- 0: Instruction cache is not reset
- 1: Instruction cache is reset

This bit can be written only when the I cache is disabled.

Bit 10 **DCEN:** Data cache enable

- 0: Data cache is disabled
- 1: Data cache is enabled

- Bit 9 **ICEN:** Instruction cache enable
0: Instruction cache is disabled
1: Instruction cache is enabled
- Bit 8 **PRFTEN:** Prefetch enable
0: Prefetch is disabled
1: Prefetch is enabled
- Bits 7:3 Reserved, must be kept cleared.
- Bits 2:0 **LATENCY:** Latency
These bits represent the ratio of the CPU clock period to the Flash memory access time.
000: Zero wait state
001: One wait state
010: Two wait states
011: Three wait states
100: Four wait states
101: Five wait states
110: Six wait states
111: Seven wait states

2.3.5 Adaptive real-time memory accelerator (ART Accelerator™)

The ART Accelerator™ is a memory accelerator which is optimized for STM32 industry-standard Cortex™-M4F processors. It balances the inherent performance advantage of the ARM Cortex™-M4F over Flash memory technologies, which normally requires the processor to wait for the Flash memory at higher operating frequencies. Thanks to the ART Accelerator™, the CPU can operate up to 168 MHz frequency without wait states, thereby increasing the overall system speed and efficiency (see [Table 3](#)).

To release the processor 210 DMIPS performance at this frequency, the accelerator implements an instruction prefetch queue and branch cache, which enables program execution from Flash memory at up to 168 MHz without wait states.

2.4 Boot configuration

Due to its fixed memory map, the code area starts from address 0x0000 0000 (accessed through the ICode/DCode buses) while the data area (SRAM) starts from address 0x2000 0000 (accessed through the system bus). The Cortex™-M4F CPU always fetches the reset vector on the ICode bus, which implies to have the boot space available only in the code area (typically, Flash memory). STM32F40x and STM32F41x microcontrollers implement a special mechanism to be able to boot from other memories (like the internal SRAM).

In the STM32F40x and STM32F41x, three different boot modes can be selected through the BOOT[1:0] pins as shown in [Table 4](#).

Table 4. Boot modes

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

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

BOOT0 is a dedicated pin while BOOT1 is shared with a GPIO pin. Once BOOT1 has been sampled, the corresponding GPIO pin is free and can be used for other purposes.

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

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

Physical remap

Once the boot pins are selected, the application software can modify the memory accessible in the code area (in this way the code can be executed through the ICode bus in place of the System bus). This modification is performed by programming the [Section 7.2.1: SYSCFG memory remap register \(SYSCFG_MEMRMP\)](#) in the SYSCFG controller.

The following memories can thus be remapped:

- Main Flash memory
- System memory
- Embedded SRAM1 (112 KB)
- FSMC Bank 1 (NOR/PSRAM 1 and 2)

Table 5. Memory mapping vs. Boot mode/physical remap

Addresses	Boot/Remap in main Flash memory	Boot/Remap in embedded SRAM	Boot/Remap in System memory	Remap in FSMC
0x2001 C000 - 0x2001 FFFF	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)
0x2000 0000 - 0x2001 BFFF	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)
0x1FFF 0000 - 0x1FFF 77FF	System memory	System memory	System memory	System memory
0x0810 0000 - 0x0FFF FFFF	Reserved	Reserved	Reserved	Reserved
0x0800 0000 - 0x080F FFFF	Flash memory	Flash memory	Flash memory	Flash memory

Table 5. Memory mapping vs. Boot mode/physical remap (continued)

Addresses	Boot/Remap in main Flash memory	Boot/Remap in embedded SRAM	Boot/Remap in System memory	Remap in FSMC
0x0400 0000 - 0x07FF FFFF	Reserved	Reserved	Reserved	FSMC Bank1 NOR/PSRAM 2 (Aliased)
0x0000 0000 - 0x03FF FFFF ⁽¹⁾⁽²⁾	Flash (1 MB) Aliased	SRAM1 (112 KB) Aliased	System memory (30 KB) Aliased	FSMC Bank1 NOR/PSRAM 1 (Aliased)

- When the FSMC is remapped at address 0x0000 0000, only the first two regions of Bank 1 memory controller (Bank1 NOR/PSRAM 1 and NOR/PSRAM 2) can be remapped. In remap mode, the CPU can access the external memory via ICode bus instead of System bus which boosts up the performance.
- Even when aliased in the boot memory space, the related memory is still accessible at its original memory space.

Embedded bootloader

The embedded bootloader mode is used to reprogram the Flash memory using one of the following serial interfaces:

- USART1(PA9/PA10)
- USART3(PB10/11 and PC10/11)
- CAN2(PB5/13)
- USB OTG FS(PA11/12) in Device mode (DFU: device firmware upgrade).

The USART peripherals operate at the internal 16 MHz oscillator (HSI) frequency, while the CAN and USB OTG FS require an external clock (HSE) multiple of 1 MHz (ranging from 4 to 26 MHz).

The embedded bootloader code is located in system memory. It is programmed by ST during production. For additional information, refer to application note AN2606.

3 CRC calculation unit

This section applies to the whole STM32F40x and STM32F41x family, unless otherwise specified.

3.1 CRC introduction

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

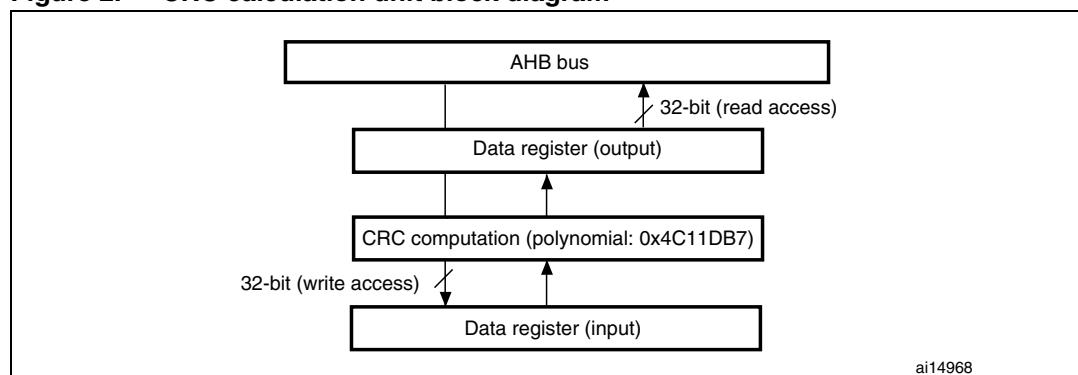
Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the EN/IEC 60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared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

3.2 CRC main features

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

The block diagram is shown in [Figure 2](#).

Figure 2. CRC calculation unit block diagram



3.3 CRC functional description

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

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

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

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

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

3.4 CRC registers

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

3.4.1 Data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bits 31:0 Data register bits

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

3.4.2 Independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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
IDR[7:0]															
Reserved								rw							

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

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

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

3.4.3 Control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bit 0 RESET bit

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

3.4.4 CRC register map

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

Table 6. CRC calculation unit register map and reset values

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

4 Power control (PWR)

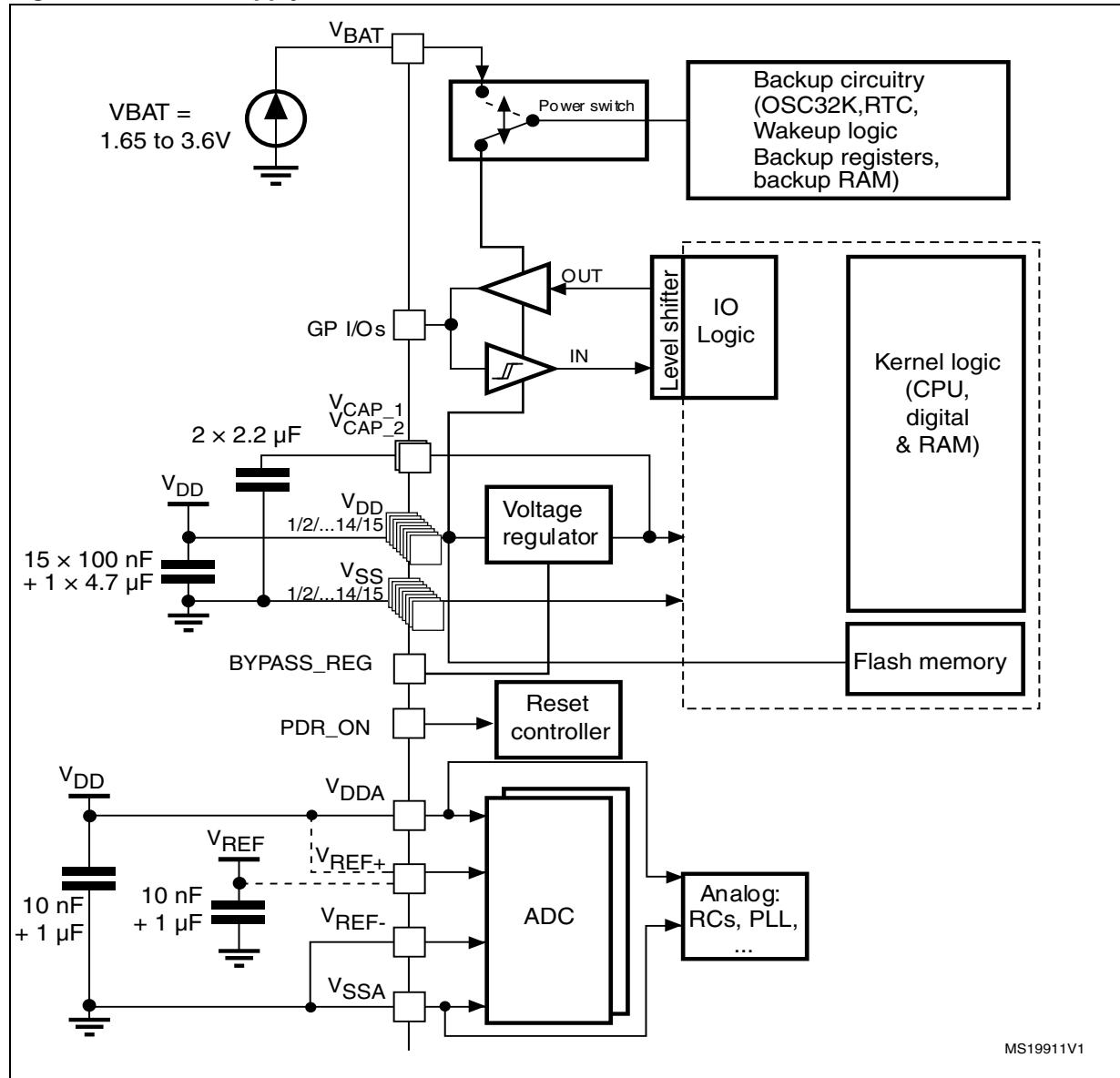
4.1 Power supplies

The device requires a 1.8-to-3.6 V operating voltage supply (V_{DD}). An embedded linear voltage regulator is used to supply the internal 1.2 V digital power.

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

Note: *Depending on the operating power supply range, some peripheral may be used with limited functionality and performance. For more details refer to section "General operating conditions" in STM32F4xx datasheets.*

Figure 3. Power supply overview



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

4.1.1 Independent A/D converter supply and reference voltage

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

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

To ensure a better accuracy of low voltage inputs, the user can connect a separate external reference voltage ADC input on V_{REF} . The voltage on V_{REF} ranges from 1.8 V to V_{DDA} .

4.1.2 Battery backup domain

Backup domain description

To retain the content of the RTC backup registers, backup SRAM, and supply the RTC 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.

To allow the RTC to operate even when the main digital supply (V_{DD}) is turned off, the V_{BAT} pin powers the following blocks:

- The RTC
- The LSE oscillator
- The backup SRAM when the low power backup regulator is enabled
- PC13 to PC15 I/Os, plus PI8 I/O (when available)

The switch to the V_{BAT} supply is controlled by the power-down reset embedded in the Reset block.

Warning: During $t_{RSTTEMPO}$ (temporization at V_{DD} startup) or after a PDR is detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT} .

During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (Refer to the datasheet for the value of $t_{RSTTEMPO}$) and $V_{DD} > V_{BAT} + 0.6$ V, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}).

If the power supply/battery connected to the V_{BAT} pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the V_{BAT} pin.

If no external battery is used in the application, it is recommended to connect V_{BAT} externally to V_{DD} through a 100 nF external ceramic capacitor.

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

- PC14 and PC15 can be used as either GPIO or LSE pins
- PC13 can be used as a GPIO or as the RTC_AF1 pin (refer to [Table 16: RTC_AF1 pin](#) for more details about this pin configuration)

Note:

Due to the fact that the switch only sinks a limited amount of current (3 mA), the use of GPIOs PC13 to PC15 are restricted: only one I/O at a time can be used as an output, 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 backup domain is supplied by V_{BAT} (analog switch connected to V_{BAT} because V_{DD} is not present), the following functions are available:

- PC14 and PC15 can be used as LSE pins only
- PC13 can be used as the RTC_AF1 pin (refer to [Table 16: RTC_AF1 pin](#) for more details about this pin configuration)

Backup domain access

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

- Access to the RTC and RTC backup registers
 - 1. Enable the power interface clock by setting the PWREN bits in the *RCC APB1 peripheral clock enable register (RCC_APB1ENR)*
 - 2. Set the DBP bit in the *PWR power control register (PWR_CR)* to enable access to the backup domain
 - 3. Select the RTC clock source: see *Section 5.2.8: RTC/AWU clock*
 - 4. Enable the RTC clock by programming the RTCEN [15] bit in the *RCC Backup domain control register (RCC_BDCR)*
- Access to the backup SRAM
 - 1. Enable the power interface clock by setting the PWREN bits in the *RCC APB1 peripheral clock enable register (RCC_APB1ENR)*
 - 2. Set the DBP bit in the *PWR power control register (PWR_CR)* to enable access to the backup domain
 - 3. Enable the backup SRAM clock by setting BKPSRAMEN bit in the *RCC APB1 peripheral clock enable register (RCC_APB1ENR)*

RTC and RTC backup registers

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar, two programmable alarm interrupts, and a periodic programmable wakeup flag with interrupt capability. The RTC contains 20 backup data registers (80 bytes) which are reset when a tamper detection event occurs. For more details refer to *Section 22: Real-time clock (RTC)*.

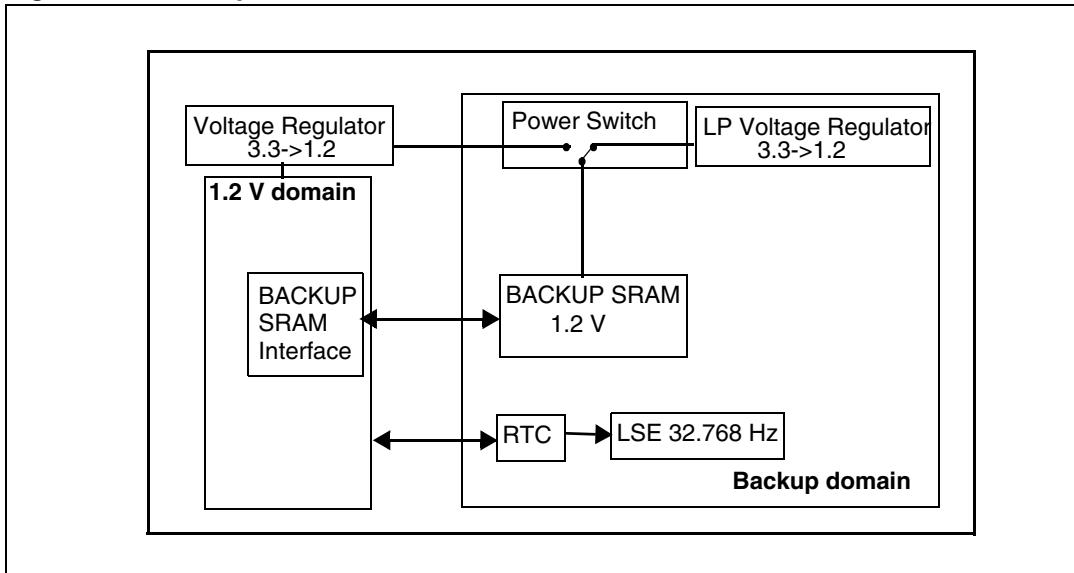
Backup SRAM

The backup domain includes 4 Kbytes of backup SRAM accessible only from the CPU, and address in 32-bit, 16-bit or 8-bit mode. Its content is retained even in Standby or V_{BAT} mode when the low power backup regulator is enabled. It can be considered as an internal EEPROM when V_{BAT} is always present.

When the backup domain is supplied by V_{DD} (analog switch connected to V_{DD}), the backup SRAM is powered from V_{DD} which replaces the V_{BAT} power supply to save battery life.

When the backup domain is supplied by V_{BAT} (analog switch connected to V_{BAT} because V_{DD} is not present), the backup SRAM is powered by a dedicated low power regulator. This regulator can be ON or OFF depending whether the application needs the backup SRAM function in Standby and V_{BAT} modes or not. The power down of this regulator is controlled by a dedicated bit, the BRE control bit of the PWR_CSR register (see *Section 4.4.2: PWR power control/status register (PWR_CSR)*).

The backup SRAM is not mass erased by a tamper event. It is read protected to prevent confidential data, such as cryptographic private key, from being accessed. The backup SRAM can be erased only through the Flash interface when a protection level change from level 1 to level 0 is requested. Refer to the description of Read protection (RDP) in the Flash programming manual.

Figure 4. Backup domain

4.1.3 Voltage regulator

An embedded linear voltage regulator supplies all the digital circuitries except for the backup domain and the Standby circuitry. The regulator output voltage is around 1.2 V.

This voltage regulator requires two external capacitors to be connected to two dedicated pins, V_{CAP_1} and V_{CAP_2} available in all packages. Specific pins must be connected either to V_{SS} or V_{DD} to activate or deactivate the voltage regulator. These pins depend on the package.

When activated by software, 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 1.2 V domain (core, memories and digital peripherals). In this mode, the regulator output voltage (around 1.2 V) can be scaled by software to different voltage values (scale 1 or scale 2 configured through the VOS bit of the PWR_CR register). The voltage scaling allows optimizing the power consumption when the device is clocked below the maximum system frequency (see [Section 4.4.1: PWR power control register \(PWR_CR\)](#)).
- In Stop mode the regulator supplies low power to the 1.2 V domain, preserving the content of registers and internal SRAM.
- In Standby mode, the regulator is powered down. The content of the registers and SRAM are lost except for the Standby circuitry and the backup domain.

Note: *For more details, refer to the voltage regulator section in the STM32F40x and STM32F41x datasheets.*

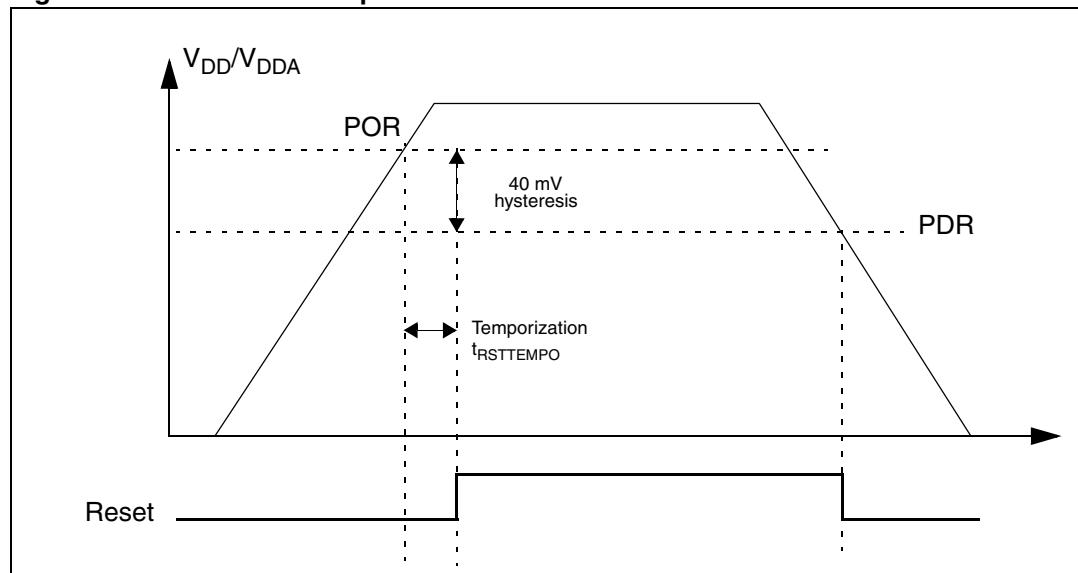
4.2 Power supply supervisor

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

The device has an integrated POR/PDR circuitry that allows proper operation starting from 1.8 V.

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

Figure 5. Power-on reset/power-down reset waveform



4.2.2 Brownout reset (BOR)

During power on, the Brownout reset (BOR) keeps the device under reset until the supply voltage reaches the specified V_{BOR} threshold.

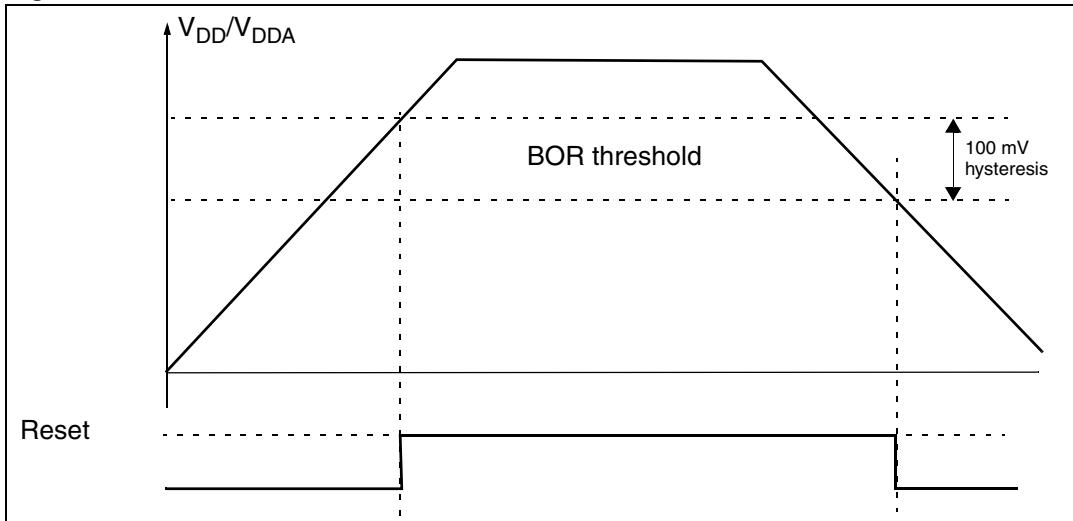
V_{BOR} is configured through device option bytes. By default, BOR is off. 4 programmable V_{BOR} thresholds can be selected.

- BOR off (V_{BOR0}): reset threshold level for 1.8 to 2.10 V voltage range
- BOR Level 1 (V_{BOR1}): reset threshold level for 2.10 to 2.40 V voltage range
- BOR Level 2 (V_{BOR2}): reset threshold level for 2.40 to 2.70 V voltage range
- BOR Level 3 (V_{BOR3}): reset threshold level for 2.70 to 3.60 V voltage range

When the supply voltage (V_{DD}) drops below the selected V_{BOR} threshold, a device reset is generated.

BOR can be disabled by programming the device option bytes. To disable the BOR function, V_{DD} must have been higher than V_{BOR0} to start the device option byte programming sequence. The power down is then monitored by the PDR (see [Section 4.2.1: Power-on reset \(POR\)/power-down reset \(PDR\)](#))

The BOR threshold hysteresis is ~100 mV (between the rising and the falling edge of the supply voltage).

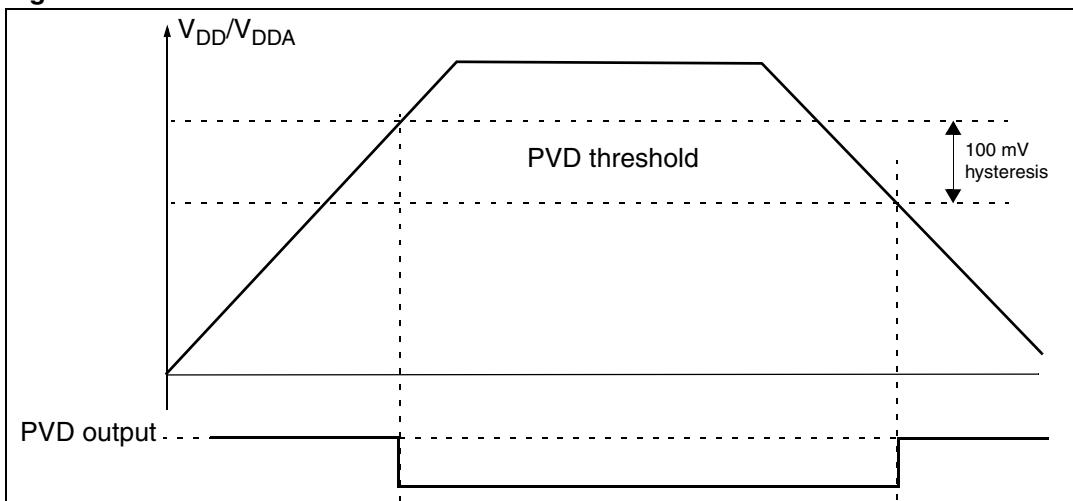
Figure 6. BOR thresholds

4.2.3 Programmable voltage detector (PVD)

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

The PVD is enabled by setting the PVDE bit.

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

Figure 7. PVD thresholds

4.3 Low-power modes

By default, the microcontroller is in Run mode after a system or a power-on reset. In Run mode the CPU is clocked by HCLK and the program code is executed. 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 devices feature three low-power modes:

- Sleep mode (Cortex™-M4F core stopped, peripherals kept running)
- Stop mode (all clocks are stopped)
- Standby mode (1.2 V domain powered off)

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

- Slowing down the system clocks
- Gating the clocks to the APBx and AHBx peripherals when they are unused.

Table 7. Low-power mode summary

Mode name	Entry	Wakeup	Effect on 1.2 V domain clocks	Effect on VDD domain clocks	Voltage regulator
Sleep (Sleep now or Sleep-on-exit)	WFI	Any interrupt	CPU CLK 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, internal and external lines)	All 1.2 V domain clocks OFF	HSI and HSE oscillators OFF	ON or in low-power mode (depends on <i>PWR power control register (PWR_CR)</i>)
Standby	PDDS bit + SLEEPDEEP bit + WFI or WFE	WKUP pin rising edge, RTC alarm (Alarm A or Alarm B), RTC Wakeup event, RTC tamper events, RTC time stamp event, external reset in NRST pin, IWDG reset			

4.3.1 Slowing down system clocks

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

For more details refer to [Section 5.3.3: RCC clock configuration register \(RCC_CFGR\)](#).

4.3.2 Peripheral clock gating

In Run mode, the HCLK_x and PCLK_x 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 AHB1 peripheral clock enable register (RCC_AHB1ENR), AHB2 peripheral clock enable register (RCC_AHB2ENR), AHB3 peripheral clock enable register (RCC_AHB3ENR) (see [RCC APB1 peripheral clock enable register \(RCC_APB1ENR\)](#) and [RCC APB2 peripheral clock enable register \(RCC_APB2ENR\)](#)).

Disabling the peripherals clocks in Sleep mode can be performed automatically by resetting the corresponding bit in RCC_AHBxLPENR and RCC_APBxLPENR registers.

4.3.3 Sleep mode

Entering Sleep mode

The Sleep mode is entered by executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the Cortex™-M4F 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.

Refer to [Table 8](#) and [Table 9](#) 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™-M4F 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 8](#) and [Table 9](#) for more details on how to exit Sleep mode.

Table 8. 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™-M4F System Control register.
Mode exit	If WFI was used for entry: Interrupt: Refer to Table 30: Vector table If WFE was used for entry Wakeup event: Refer to Section 9.2.3: Wakeup event management
Wakeup latency	None

Table 9. Sleep-on-exit

Sleep-on-exit	Description
Mode entry	WFI (wait for interrupt) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Refer to the Cortex™-M4F System Control register.
Mode exit	Interrupt: refer to Table 30: Vector table .
Wakeup latency	None

4.3.4 Stop mode

The Stop mode is based on the Cortex™-M4F deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode. In Stop mode, all clocks in the 1.2 V domain are stopped, the PLLs, the HSI and the HSE RC oscillators are disabled. Internal SRAM and register contents are preserved.

By setting the FPDS bit in the PWR_CR register, the Flash memory also enters power down mode when the device enters Stop mode. When the Flash memory is in power down mode, an additional startup delay is incurred when waking up from Stop mode.

Entering Stop mode

Refer to [Table 10](#) 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 [PWR 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 17.3 in Section 17: Independent watchdog \(IWDG\)](#).
- Real-time clock (RTC): this is configured by the RTCEN bit in the [RCC Backup domain control register \(RCC_BDCR\)](#)
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the [RCC clock control & status register \(RCC_CSR\)](#).
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the [RCC Backup domain control register \(RCC_BDCR\)](#).

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

Exiting Stop mode

Refer to [Table 10](#) 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 10. 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™-M4F System Control register – Clear PDDE bit in Power Control register (PWR_CR) – Select the voltage regulator mode by configuring LPDS bit in PWR_CR <p>Note: To enter the Stop mode, all EXTI Line pending bits (in Pending register (EXTI_PR)), the RTC Alarm (Alarm A and Alarm B), RTC wakeup, RTC tamper, and RTC time stamp flags, must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.</p>
Mode exit	<p>If WFI was used for entry: All EXTI lines configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). Refer to Table 30: Vector table on page 195.</p> <p>If WFE was used for entry: All EXTI Lines configured in event mode. Refer to Section 9.2.3: Wakeup event management on page 200</p>
Wakeup latency	HSI RC wakeup time + regulator wakeup time from Low-power mode

4.3.5 Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the Cortex™-M4F deepsleep mode, with the voltage regulator disabled. The 1.2 V domain is consequently powered off. The PLLs, the HSI oscillator and the HSE oscillator are also switched off. SRAM and register contents are lost except for registers in the backup domain

(RTC registers, RTC backup register and backup SRAM), and Standby circuitry (see [Figure 3](#)).

Entering Standby mode

Refer to [Table 11](#) 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 17.3 in Section 17: Independent watchdog \(IWDG\)](#).
- Real-time clock (RTC): this is configured by the RTCEN bit in the backup domain control register (RCC_BDCR)
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the Control/status register (RCC_CSR).
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the backup domain control register (RCC_BDCR)

Exiting Standby mode

The microcontroller exits Standby mode when an external Reset (NRST pin), an IWDG Reset, a rising edge on WKUP pin, an RTC alarm, a tamper event, or a time stamp event is detected. All registers are reset after wakeup from Standby except for [PWR 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 [PWR power control/status register \(PWR_CSR\)](#) indicates that the MCU was in Standby mode.

Refer to [Table 11](#) for more details on how to exit Standby mode.

Table 11. Standby mode

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP in Cortex™-M4F System Control register – Set PDDS bit in Power Control register (PWR_CR) – Clear WUF bit in Power Control/Status register (PWR_CSR) – Clear the RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup, Tamper or Timestamp flags)
Mode exit	WKUP pin rising edge, RTC alarm (Alarm A and Alarm B), RTC wakeup, tamper event, time stamp event, 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 for:

- Reset pad (still available)
- RTC_AF1 pin (PC13) if configured for tamper, time stamp, RTC Alarm out, or RTC clock calibration out
- WKUP pin (PA0), 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™-M4F core is no longer clocked.

However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged even when using the low-power modes extensively. For more details, refer to [Section 32.16.1: Debug support for low-power modes](#).

4.3.6 Programming the RTC alternate functions to wake up the device from the Stop and Standby modes

The MCU can be woken up from a low-power mode by an RTC alternate function.

The RTC alternate functions are the RTC alarms (Alarm A and Alarm B), RTC wakeup, RTC tamper event detection and RTC time stamp event detection.

These RTC alternate functions can wake up the system from the Stop and Standby low-power modes.

The system can also wake up from low-power modes without depending on an external interrupt (Auto-wakeup mode), by using the RTC alarm or the RTC wakeup events.

The RTC provides a programmable time base for waking up from the Stop or Standby mode at regular intervals.

For this purpose, two of the three alternate RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the [*RCC Backup domain control register \(RCC_BDCR\)*](#):

- Low-power 32.768 kHz external crystal oscillator (LSE OSC)
This clock source provides a precise time base with a very low-power consumption (additional consumption of less than 1 µA under 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 use minimum power.

RTC alternate functions to wake up the device from the Stop mode

- To wake up the device from the Stop mode with an RTC alarm event, it is necessary to:
 - a) Configure the EXTI Line 17 to be sensitive to rising edges (Interrupt or Event modes)
 - b) Enable the RTC Alarm Interrupt in the RTC_CR register
 - c) Configure the RTC to generate the RTC alarm
- To wake up the device from the Stop mode with an RTC tamper or time stamp event, it is necessary to:
 - a) Configure the EXTI Line 21 to be sensitive to rising edges (Interrupt or Event modes)
 - b) Enable the RTC time stamp Interrupt in the RTC_CR register or the RTC tamper interrupt in the RTC_TAFCR register
 - c) Configure the RTC to detect the tamper or time stamp event
- To wake up the device from the Stop mode with an RTC wakeup event, it is necessary to:
 - a) Configure the EXTI Line 22 to be sensitive to rising edges (Interrupt or Event modes)
 - b) Enable the RTC wakeup interrupt in the RTC_CR register
 - c) Configure the RTC to generate the RTC Wakeup event

RTC alternate functions to wake up the device from the Standby mode

- To wake up the device from the Standby mode with an RTC alarm event, it is necessary to:
 - a) Enable the RTC alarm interrupt in the RTC_CR register
 - b) Configure the RTC to generate the RTC alarm
- To wake up the device from the Standby mode with an RTC tamper or time stamp event, it is necessary to:
 - a) Enable the RTC time stamp interrupt in the RTC_CR register or the RTC tamper interrupt in the RTC_TAFCR register
 - b) Configure the RTC to detect the tamper or time stamp event
- To wake up the device from the Standby mode with an RTC wakeup event, it is necessary to:
 - a) Enable the RTC wakeup interrupt in the RTC_CR register
 - b) Configure the RTC to generate the RTC wakeup event

Safe RTC alternate function wakeup flag clearing sequence

If the selected RTC alternate function is set before the PWR wakeup flag (WUTF) is cleared, it will not be detected on the next event as detection is made once on the rising edge.

To avoid bouncing on the pins onto which the RTC alternate functions are mapped, and exit correctly from the Stop and Standby modes, it is recommended to follow the sequence below before entering the Standby mode:

- When using RTC alarm to wake up the device from the low-power modes:
 - a) Disable the RTC alarm interrupt (ALRAIE or ALRBIE bits in the RTC_CR register)
 - b) Clear the RTC alarm (ALRAF/ALRBF) flag
 - c) Clear the PWR Wakeup (WUF) flag
 - d) Enable the RTC alarm interrupt
 - e) Re-enter the low-power mode
- When using RTC wakeup to wake up the device from the low-power modes:
 - a) Disable the RTC Wakeup interrupt (WUTIE bit in the RTC_CR register)
 - b) Clear the RTC Wakeup (WUTF) flag
 - c) Clear the PWR Wakeup (WUF) flag
 - d) Enable the RTC Wakeup interrupt
 - e) Re-enter the low power mode
- When using RTC tamper to wake up the device from the low-power modes:
 - a) Disable the RTC tamper interrupt (TAMPIE bit in the RTC_TAFCR register)
 - b) Clear the Tamper (TAMP1F/TSF) flag
 - c) Clear the PWR Wakeup (WUF) flag
 - d) Enable the RTC tamper interrupt
 - e) Re-enter the low-power mode
- When using RTC time stamp to wake up the device from the low-power modes:
 - a) Disable the RTC time stamp interrupt (TSIE bit in RTC_CR)
 - b) Clear the RTC time stamp (TSF) flag
 - c) Clear the PWR Wakeup (WUF) flag
 - d) Enable the RTC TimeStamp interrupt
 - e) Re-enter the low-power mode

4.4 Power control registers

4.4.1 PWR power control register (PWR_CR)

Address offset: 0x00

Reset value: 0x0000 4000 (reset by wakeup from Standby mode)

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

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

Bit 14 **VOS**: Regulator voltage scaling output selection

This bit controls the main internal voltage regulator output voltage to achieve a tradeoff between performance and power consumption when the device does not operate at the maximum frequency (refer to the datasheets for more details).

0: Scale 2 mode

1: Scale 1 mode (default value at reset)

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

Bit 9 **FPDS**: Flash power down in Stop mode

When set, the Flash memory enters power down mode when the device enters Stop mode. This allows to achieve a lower consumption in stop mode but a longer restart time.

0: Flash memory not in power down when the device is in Stop mode

1: Flash memory in power down when the device is in Stop mode

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

In reset state, the RCC_BDCR register, the RTC registers (including the backup registers), and the BRE bit of the PWR_CSR register, are protected against parasitic write access. This bit must be set to enable write access to these registers.

0: Access to RTC and RTC Backup registers and backup SRAM disabled

1: Access to RTC and RTC Backup registers and backup SRAM enabled

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: 2.0 V

001: 2.1 V

010: 2.3 V

011: 2.5 V

100: 2.6 V

101: 2.7 V

110: 2.8 V

111: 2.9 V

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

Bit 4 PVDE: Power voltage detector enable

This bit is set and cleared by software.

0: PVD disabled

1: PVD enabled

Bit 3 CSBF: Clear standby flag

This bit is always read as 0.

0: No effect

1: Clear the SBF Standby Flag (write).

Bit 2 CWUF: Clear wakeup flag

This bit is always read as 0.

0: No effect

1: Clear the WUF Wakeup Flag **after 2 System clock cycles**

Bit 1 PDSS: 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 deep sleep

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

0: Voltage regulator on during Stop mode

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

4.4.2 PWR power control/status register (PWR_CSR)

Address offset: 0x04

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

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

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

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

Bit 14 VOSRDY: Regulator voltage scaling output selection ready bit

0: Not ready

1: Ready

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

Bit 9 BRE: Backup regulator enable

When set, the Backup regulator (used to maintain backup SRAM content in Standby and V_{BAT} modes) is enabled. If BRE is reset, the backup regulator is switched off. The backup SRAM can still be used but its content will be lost in the Standby and V_{BAT} modes. Once set, the application must wait that the Backup Regulator Ready flag (BRR) is set to indicate that the data written into the RAM will be maintained in the Standby and V_{BAT} modes.

- 0: Backup regulator disabled
- 1: Backup regulator enabled

Note: This bit is not reset when the device wakes up from Standby mode, by a system reset, or by a power reset.

Bit 8 EWUP: Enable WKUP pin

This bit is set and cleared by software.

- 0: WKUP pin is used for general purpose I/O. An event on the WKUP pin does not wakeup the device from Standby mode.
- 1: WKUP pin is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP pin wakes-up the system from Standby mode).

Note: This bit is reset by a system reset.

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

Bit 3 BRR: Backup regulator ready

Set by hardware to indicate that the Backup Regulator is ready.

- 0: Backup Regulator not ready
- 1: Backup Regulator ready

Note: This bit is not reset when the device wakes up from Standby mode or by a system reset or power reset.

Bit 2 PVDO: PVD output

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

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

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

Bit 1 SBF: Standby flag

This bit is set by hardware and cleared only by a POR/PDR (power-on reset/power-down reset) or by setting the CSBF bit in the [PWR power control register \(PWR_CR\)](#)

- 0: Device has not been in Standby mode
- 1: Device has been in Standby mode

Bit 0 WUF: Wakeup flag

This bit is set by hardware and cleared only by a POR/PDR (power-on reset/power-down reset) or by setting the CWUF bit in the [PWR 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 (Alarm A or Alarm B), RTC Tamper event, RTC TimeStamp event or RTC Wakeup).

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

4.4.3 PWR register map

The following table summarizes the PWR registers.

Table 12. 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		
																		VOS					PLS[2:0]			PVDE		CSBF		CWUF		PDDS		LPDS	
0x000	PWR_CR Reset value																	1	Reserved				0 0 0	0	0	0	0	0	0	0	0	0	0	0	
0x004	PWR_CSR Reset value																	0	VOSRDY				Reserve	0	0	0	0	0	0	0	0	0	0	0	

Refer to [Table 1 on page 50](#) for the register boundary addresses.

5 Reset and clock control (RCC)

5.1 Reset

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

5.1.1 System reset

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

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

1. A low level on the NRST pin (external reset)
2. Window watchdog end of count condition (WWDG reset)
3. Independent watchdog end of count condition (IWDG reset)
4. A software reset (SW reset) (see [Software reset](#))
5. Low-power management reset (see [Low-power management reset](#))

Software reset

The reset source can be identified by checking the reset flags in the [RCC clock control & status register \(RCC_CSR\)](#).

The SYSRESETREQ bit in Cortex™-M4F Application Interrupt and Reset Control Register must be set to force a software reset on the device. Refer to the Cortex™-M4F technical reference manual for more details.

Low-power management reset

There are two ways of generating a low-power management reset:

1. Reset generated when entering the Standby mode:

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

2. Reset when entering the Stop mode:

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

For further information on the user option bytes, refer to the STM32F40x and STM32F41x Flash programming manual available from your ST sales office.

5.1.2 Power reset

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

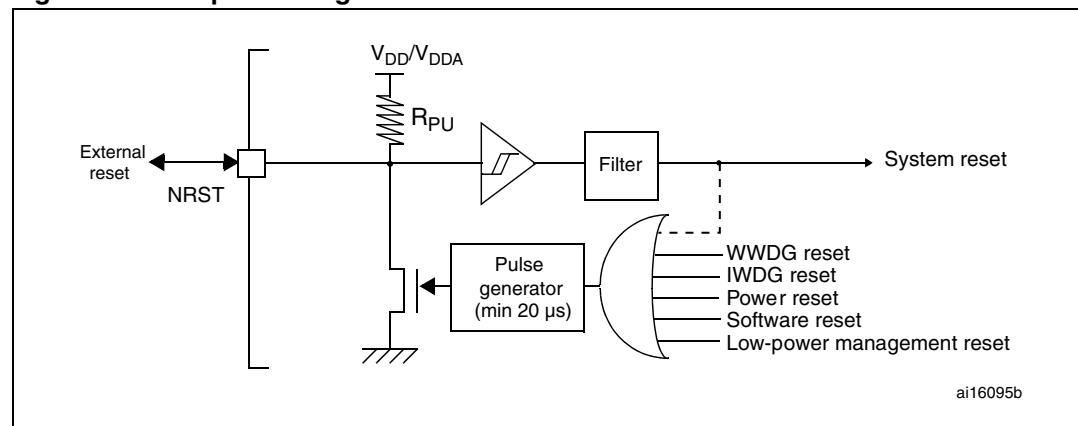
1. Power-on/power-down reset (POR/PDR reset) or brownout (BOR) reset
2. When exiting the Standby mode

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

These sources act on the NRST pin and it is always kept low during the delay phase. The RESET service routine vector is fixed at address 0x0000_0004 in the memory map.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each reset source (external or internal reset). In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

Figure 8. Simplified diagram of the reset circuit



The Backup domain has two specific resets that affect only the Backup domain (see [Figure 4](#)).

5.1.3 Backup domain reset

The backup domain reset sets all RTC registers and the RCC_BDCR register to their reset values. The BKPSRAM is not affected by this reset. The only way of resetting the BKPSRAM is through the Flash interface by requesting a protection level change from 1 to 0.

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

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

5.2 Clocks

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

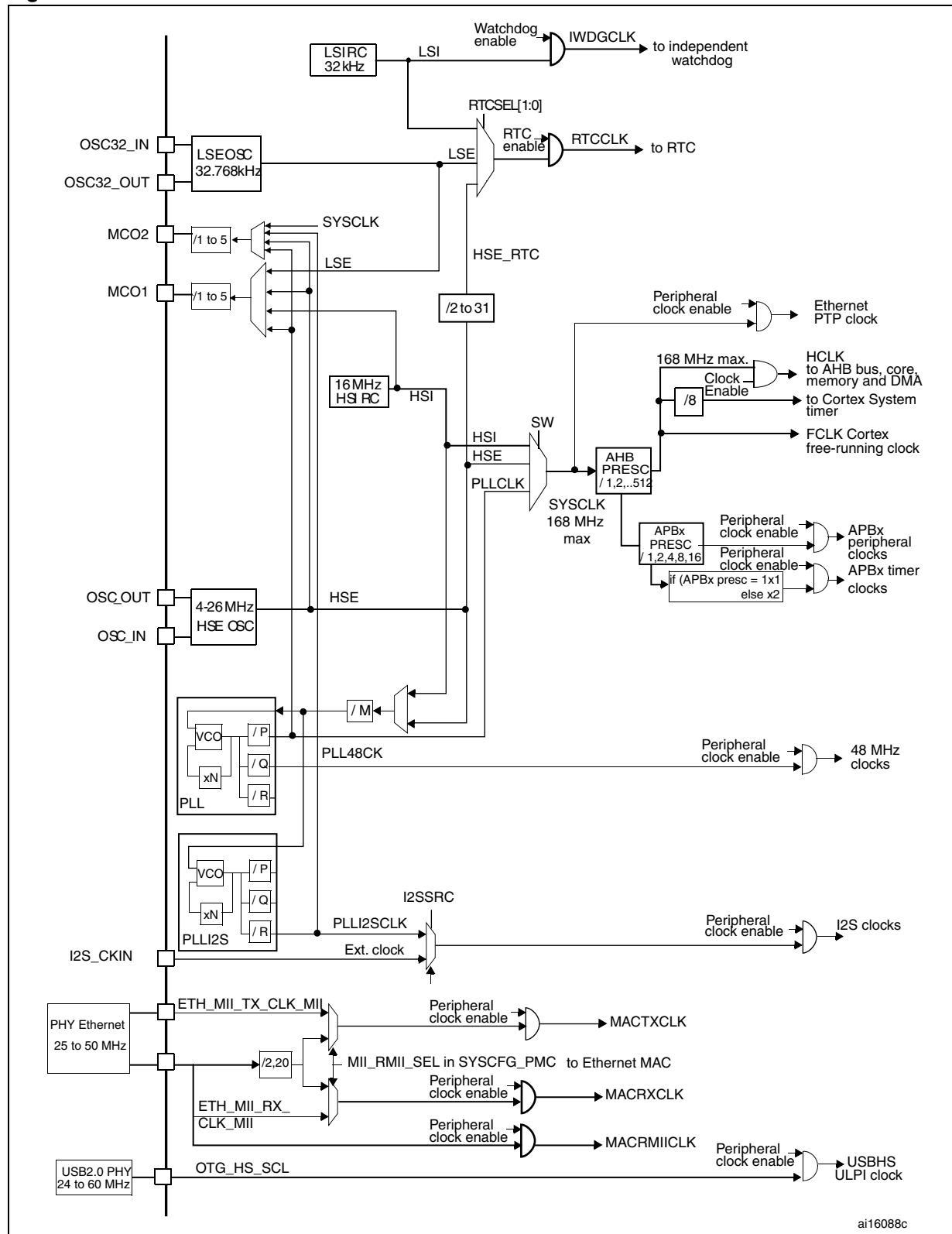
- HSI oscillator clock
- HSE oscillator clock
- Main PLL (PLL) clock

The devices have the two following secondary clock sources:

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

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

Figure 9. Clock tree



- For full details about the internal and external clock source characteristics, refer to the Electrical characteristics section in the device datasheet

The clock controller provides a high degree of flexibility to the application in the choice of the external crystal or the oscillator to run the core and peripherals at the highest frequency and, guarantee the appropriate frequency for peripherals that need a specific clock like Ethernet, USB OTG FS and HS, I2S and SDIO.

Several prescalers are used to configure the AHB frequency, the high-speed APB (APB2) and the low-speed APB (APB1) domains. The maximum frequency of the AHB domain is 168 MHz. The maximum allowed frequency of the high-speed APB2 domain is 84 MHz. The maximum allowed frequency of the low-speed APB1 domain is 42 MHz

All peripheral clocks are derived from the system clock (SYSCLK) except for:

- The USB OTG FS clock (48 MHz), the random analog generator (RNG) clock (≤ 48 MHz) and the SDIO clock (≤ 48 MHz) which are coming from a specific output of PLL (PLL48CLK)
- The I2S clock

To achieve high-quality audio performance, the I2S clock can be derived either from a specific PLL (PLLI2S) or from an external clock mapped on the I2S_CKIN pin. For more information about I2S clock frequency and precision, refer to [Section 25.4.4: Clock generator](#).

- The USB OTG HS (60 MHz) clock which is provided from the external PHY
- The Ethernet MAC clocks (TX, RX and RMII) which are provided from the external PHY. For further information on the Ethernet configuration, please refer to [Section 28.4.4: MII/RMII selection](#) in the Ethernet peripheral description. When the Ethernet is used, the AHB clock frequency must be at least 25 MHz.

The RCC feeds the external clock of the Cortex System Timer (SysTick) with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or with the Cortex clock (HCLK), configurable in the SysTick control and status register.

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

1. If the APB prescaler is 1, the timer clock frequencies are set to the same frequency as that of the APB domain to which the timers are connected.
2. Otherwise, they are set to twice ($\times 2$) the frequency of the APB domain to which the timers are connected.

FCLK acts as CortexTM-M4F free-running clock. For more details, refer to the CortexTM-M4F technical reference manual.

5.2.1 HSE clock

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

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

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

Figure 10. HSE/ LSE clock sources

Hardware configuration	
External clock	
Crystal/ceramic resonators	

External source (HSE bypass)

In this mode, an external clock source must be provided. You select this mode by setting the HSEBYP and HSEON bits in the [RCC clock control register \(RCC_CR\)](#). The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC_IN pin while the OSC_OUT pin should be left hi-Z. See [Figure 10](#).

External crystal/ceramic resonator (HSE crystal)

The HSE has the advantage of producing a very accurate rate on the main clock.

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

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

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

5.2.2 HSI clock

The HSI clock signal is generated from an internal 16 MHz RC oscillator and can be used directly as a system clock, or 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 [*RCC 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 [*RCC clock control register \(RCC_CR\)*](#).

The HSIRDY flag in the [*RCC 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 [*RCC 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 5.2.7: Clock security system \(CSS\) on page 89*](#).

5.2.3 PLL configuration

The STM32F4xx devices feature two PLLs:

- A main PLL (PLL) clocked by the HSE or HSI oscillator and featuring two different output clocks:
 - The first output is used to generate the high speed system clock (up to 168 MHz)
 - The second output is used to generate the clock for the USB OTG FS (48 MHz), the random analog generator (<48 MHz) and the SDIO (<48 MHz).
- A dedicated PLL (PLLI2S) used to generate an accurate clock to achieve high-quality audio performance on the I2S interface.

Since the main-PLL configuration parameters cannot be changed once PLL is enabled, it is recommended to configure PLL before enabling it (selection of the HSI or HSE oscillator as PLL clock source, and configuration of division factors M, N, P, and Q).

The PLLI2S uses the same input clock as PLL (PLLM[5:0] and PLLSRC bits are common to both PLLs). However, the PLLI2S has dedicated enable/disable and division factors (N and R) configuration bits. Once the PLLI2S is enabled, the configuration parameters cannot be changed.

The two PLLs are disabled by hardware when entering Stop and Standby modes, or when an HSE failure occurs when HSE or PLL (clocked by HSE) are used as system clock. [*RCC PLL configuration register \(RCC_PLLCFGR\)*](#) and [*RCC clock configuration register \(RCC_CFGR\)*](#) can be used to configure PLL and PLLI2S, respectively.

5.2.4 LSE clock

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

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

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

External source (LSE bypass)

In this mode, an external clock source must be provided. It must have a frequency up to 1 MHz. You select this mode by setting the LSEBYP and LSEON bits in the *RCC Backup domain control register (RCC_BDCR)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin should be left Hi-Z. See *Figure 10*.

5.2.5 LSI clock

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

The LSI RC can be switched on and off using the LSION bit in the *RCC clock control & status register (RCC_CSR)*.

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

5.2.6 System clock (SYSCLK) selection

After a system reset, the HSI oscillator is selected as the system clock. When a clock source is used directly or through PLL as the 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 that is not yet ready is selected, the switch occurs when the clock source is ready. Status bits in the *RCC clock control register (RCC_CR)* indicate which clock(s) is (are) ready and which clock is currently used as the system clock.

5.2.7 Clock security system (CSS)

The 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, this oscillator is automatically disabled, a clock failure event is sent to the break inputs of advanced-control timers TIM1 and TIM8, 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 CortexTM-M4F NMI (non-maskable interrupt) exception vector.

Note: When the CSS is enabled, if the HSE clock happens to fail, the CSS generates an interrupt, which causes the automatic generation of an NMI. The NMI is executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, the application has to clear the CSS interrupt in the NMI ISR 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 meaning that it is directly used as PLL input clock, and that PLL clock is the system clock) and a failure is detected, then the system clock switches to the HSI oscillator and the HSE oscillator is disabled.

If the HSE oscillator clock was the clock source of PLL used as the system clock when the failure occurred, PLL is also disabled. In this case, if the PLLI2S was enabled, it is also disabled when the HSE fails.

5.2.8 RTC/AWU clock

Once the RTCCLK clock source has been selected, the only possible way of modifying the selection is to reset the power domain.

The RTCCLK clock source can be either the HSE 1 MHz (HSE divided by a programmable prescaler), the LSE or the LSI clock. This is selected by programming the RTCSEL[1:0] bits in the *RCC Backup domain control register (RCC_BDCR)* and the RTCPRE[4:0] bits in *RCC clock configuration register (RCC_CFGR)*. This selection cannot be modified without resetting the Backup domain.

If the LSE is selected as the RTC clock, the RTC will work normally if the backup or the system supply disappears. If the LSI is selected as the AWU clock, the AWU state is not guaranteed if the system supply disappears. If the HSE oscillator divided by a value between 2 and 31 is used as the RTC clock, the RTC state is not guaranteed if the backup or the system supply disappears.

The LSE clock is in the Backup domain, whereas the HSE and LSI clocks are not. As a consequence:

- If LSE is selected as the 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 Auto-wakeup unit (AWU) clock:
 - The AWU state is not guaranteed if the V_{DD} supply is powered off. Refer to [Section 5.2.5: LSI clock on page 89](#) for more details on LSI calibration.
- If the HSE clock 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 1.2 V domain).

Note: To read the RTC calendar register when the APB1 clock frequency is less than seven times the RTC clock frequency ($f_{APB1} < 7 \times f_{RTCLK}$), the software must read the calendar time and date registers twice. The data are correct if the second read access to RTC_TR gives the same result than the first one. Otherwise a third read access must be performed.

5.2.9 Watchdog clock

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

5.2.10 Clock-out capability

Two microcontroller clock output (MCO) pins are available:

- MCO1

You can output four different clock sources onto the MCO1 pin (PA8) using the configurable prescaler (from 1 to 5):

- HSI clock
- LSE clock
- HSE clock
- PLL clock

The desired clock source is selected using the MCO1PRE[2:0] and MCO1[1:0] bits in the *RCC clock configuration register (RCC_CFGR)*.

- MCO2

You can output four different clock sources onto the MCO2 pin (PC9) using the configurable prescaler (from 1 to 5):

- HSE clock
- PLL clock
- System clock (SYSCLK)
- PLLI2S clock

The desired clock source is selected using the MCO2PRE[2:0] and MCO2 bits in the *RCC clock configuration register (RCC_CFGR)*.

For the different MCO pins, the corresponding GPIO port has to be programmed in alternate function mode.

The selected clock to output onto MCO must not exceed 100 MHz (the maximum I/O speed).

5.2.11 Internal/external clock measurement using TIM5/TIM11

It is possible to indirectly measure the frequencies of all on-board clock source generators by means of the input capture of TIM5 channel4 and TIM11 channel1 as shown in *Figure 11* and *Figure 11*.

Internal/external clock measurement using TIM5 channel4

TIM5 has an input multiplexer which allows choosing whether the input capture is triggered by the I/O or by an internal clock. This selection is performed through the TI4_RMP [1:0] bits in the TIM5_OR register.

The primary purpose of having the LSE connected to the channel4 input capture is to be able to precisely measure the HSI (this requires to have the HSI used as the system clock source). The number of HSI clock counts between consecutive edges of the LSE signal provides a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) we can 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.

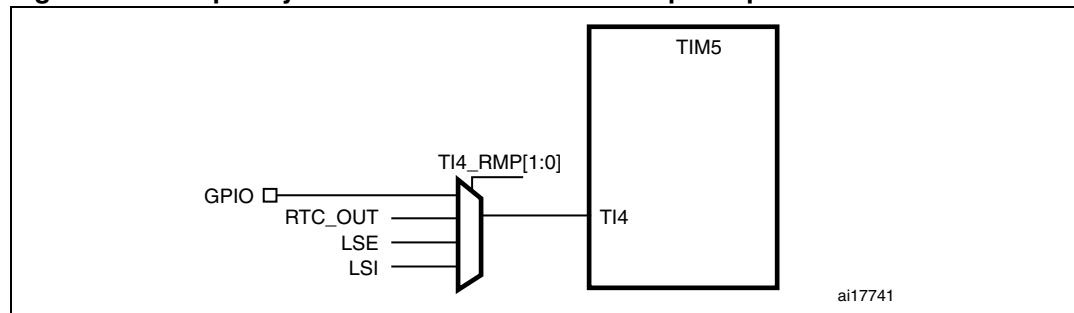
The basic concept consists in providing a relative measurement (e.g. HSI/LSE ratio): the precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio, the better the measurement.

It is also possible to measure the LSI frequency: this is useful for applications that do not have a crystal. The ultralow-power LSI oscillator has a large manufacturing process deviation: by measuring it versus the HSI clock source, it is possible to determine its frequency with the precision of the HSI. The measured value can be used to have more accurate RTC time base timeouts (when LSI is used as the RTC clock source) and/or an IWDG timeout with an acceptable accuracy.

Use the following procedure to measure the LSI frequency:

1. Enable the TIM5 timer and configure channel4 in Input capture mode.
2. Set the TI4_RMP bits in the TIM5_OR register to 0x01 to connect the LSI clock internally to TIM5 channel4 input capture for calibration purposes.
3. Measure the LSI clock frequency using the TIM5 capture/compare 4 event or interrupt.
4. Use the measured LSI frequency to update the prescaler of the RTC depending on the desired time base and/or to compute the IWDG timeout.

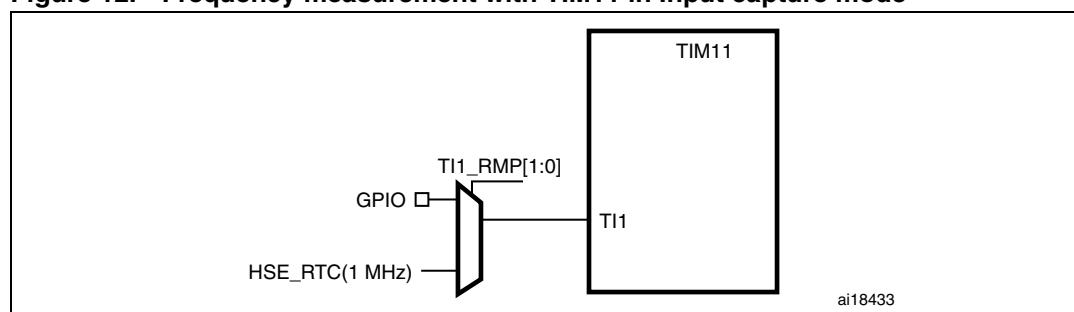
Figure 11. Frequency measurement with TIM5 in Input capture mode



Internal/external clock measurement using TIM11 channel1

TIM11 has an input multiplexer which allows choosing whether the input capture is triggered by the I/O or by an internal clock. This selection is performed through TI1_RMP [1:0] bits in the TIM11_OR register. The HSE_RTC clock (HSE divided by a programmable prescaler) is connected to channel 1 input capture to have a rough indication of the external crystal frequency. This requires that the HSI is the system clock source. This can be useful for instance to ensure compliance with the IEC 60730/IEC 61335 standards which require to be able to determine harmonic or subharmonic frequencies (-50/+100% deviations).

Figure 12. Frequency measurement with TIM11 in Input capture mode



5.3 RCC registers

Refer to [Section 1.1: List of abbreviations for registers](#) for a list of abbreviations used in register descriptions.

5.3.1 RCC clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLL2S RDY	PLL2S ON	PLLRDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
				r	rw	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		r	rw	

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

Bit 27 **PLL2SRDY**: PLLI2S clock ready flag

Set by hardware to indicate that the PLLI2S is locked.

0: PLLI2S unlocked
1: PLLI2S locked

Bit 26 **PLL2SON**: PLLI2S enable

Set and cleared by software to enable PLLI2S.

Cleared by hardware when entering Stop or Standby mode.

0: PLLI2S OFF
1: PLLI2S ON

Bit 25 **PLLRDY**: Main PLL (PLL) clock ready flag

Set by hardware to indicate that PLL is locked.

0: PLL unlocked
1: PLL locked

Bit 24 **PLLON**: Main PLL (PLL) enable

Set and cleared by software to enable PLL.

Cleared by hardware when entering Stop or Standby mode. This bit cannot be reset if PLL clock is used as 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 an oscillator failure is detected.

0: Clock security system OFF (Clock detector OFF)
1: Clock security system ON (Clock detector ON if HSE oscillator is stable, OFF if not)

Bit 18 HSEBYP: HSE clock bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit, to be used by the device.

The HSEBYP bit can be written only if the HSE oscillator is disabled.

0: HSE oscillator not bypassed

1: HSE oscillator bypassed with an external clock

Bit 17 HSERDY: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable. After the HSEON bit is cleared, HSERDY goes low after 6 HSE oscillator clock cycles.

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]: Internal high-speed clock calibration

These bits are initialized automatically at startup.

Bits 7:3 HSITRIM[4:0]: Internal high-speed clock trimming

These bits provide an additional user-programmable trimming value that is added to the HSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the internal HSI RC.

Bit 2 Reserved, must be kept at reset value.**Bit 1 HSIRDY:** Internal high-speed clock ready flag

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

0: HSI oscillator not ready

1: HSI oscillator ready

Bit 0 HSION: Internal high-speed clock enable

Set and cleared by software.

Set by hardware to force the HSI oscillator ON when leaving the Stop or Standby mode or in case of a failure of the HSE oscillator used directly or indirectly as the system clock. This bit cannot be cleared if the HSI is used directly or indirectly as the system clock.

0: HSI oscillator OFF

1: HSI oscillator ON

5.3.2 RCC PLL configuration register (RCC_PLLCFGR)

Address offset: 0x04

Reset value: 0x2400 3010

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

This register is used to configure the PLL clock outputs according to the formulas:

- $f_{(VCO\ clock)} = f_{(PLL\ clock\ input)} \times (PLLN / PLLM)$
- $f_{(PLL\ general\ clock\ output)} = f_{(VCO\ clock)} / PLLP$
- $f_{(USB\ OTG\ FS,\ SDIO,\ RNG\ clock\ output)} = f_{(VCO\ clock)} / PLLQ$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			PLLQ3	PLLQ2	PLLQ1	PLLQ0	Reserv ed	PLLSRC	Reserved			PLLP1	PLLP0		
			rw	rw	rw	rw		rw				rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserv ed	PLLN8	PLLN7	PLLN6	PLLN5	PLLN4	PLLN3	PLLN2	PLLN1	PLLN0	PLLM5	PLLM4	PLLM3	PLLM2	PLLM1	PLLM0
	rw	rw	rw	rw	rw	rw	rw	rw	rw						

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

Bits 27:24 **PLLQ**: Main PLL (PLL) division factor for USB OTG FS, SDIO and random number generator clocks

Set and cleared by software to control the frequency of USB OTG FS clock, the random number generator clock and the SDIO clock. These bits should be written only if PLL is disabled.

Caution: The USB OTG FS requires a 48 MHz clock to work correctly. The SDIO and the random number generator need a frequency lower than or equal to 48 MHz to work correctly.

USB OTG FS clock frequency = VCO frequency / PLLQ with $2 \leq PLLQ \leq 15$

0000: PLLQ = 0, wrong configuration

0001: PLLQ = 1, wrong configuration

0010: PLLQ = 2

0011: PLLQ = 3

0100: PLLQ = 4

...

1111: PLLQ = 15

Bit 23 Reserved, must be kept at reset value.

Bit 22 **PLLSRC**: Main PLL(PLL) and audio PLL (PLLI2S) entry clock source

Set and cleared by software to select PLL and PLLI2S clock source. This bit can be written only when PLL and PLLI2S are disabled.

0: HSI clock selected as PLL and PLLI2S clock entry

1: HSE oscillator clock selected as PLL and PLLI2S clock entry

Bits 21:18 Reserved, must be kept at reset value.

Bits 17:16 **PLLP:** Main PLL (PLL) division factor for main system clock

Set and cleared by software to control the frequency of the general PLL output clock. These bits can be written only if PLL is disabled.

Caution: The software has to set these bits correctly not to exceed 168 MHz on this domain.

PLL output clock frequency = VCO frequency / PLLP with PLLP = 2, 4, 6, or 8

00: PLLP = 2
01: PLLP = 4
10: PLLP = 6
11: PLLP = 8

Bits 14:6 **PLLN:** Main PLL (PLL) multiplication factor for VCO

Set and cleared by software to control the multiplication factor of the VCO. These bits can be written only when PLL is disabled. Only half-word and word accesses are allowed to write these bits.

Caution: The software has to set these bits correctly to ensure that the VCO output frequency is between 64 and 432 MHz.

VCO output frequency = VCO input frequency × PLLN with $64 \leq \text{PLLN} \leq 432$

00000000: PLLN = 0, wrong configuration
00000001: PLLN = 1, wrong configuration

...

00011111: PLLN = 63
00100000: PLLN = 64
00100001: PLLN = 65

...

01100000: PLLN = 192

...

110110000: PLLN = 432
110110001: PLLN = 433, wrong configuration

...

111111111: PLLN = 511, wrong configuration

Bits 5:0 **PLLM:** Division factor for the main PLL (PLL) and audio PLL (PLLI2S) input clock

Set and cleared by software to divide the PLL and PLLI2S input clock before the VCO. These bits can be written only when the PLL and PLLI2S are disabled.

Caution: The software has to set these bits correctly to ensure that the VCO input frequency ranges from 1 to 2 MHz. It is recommended to select a frequency of 2 MHz to limit PLL jitter.

VCO input frequency = PLL input clock frequency / PLLM with $2 \leq \text{PLLM} \leq 63$

00000: PLLM = 0, wrong configuration
000001: PLLM = 1, wrong configuration
000010: PLLM = 2
000011: PLLM = 3
000100: PLLM = 4
...
111110: PLLM = 62
111111: PLLM = 63

5.3.3 RCC clock configuration register (RCC_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSC R	MCO1		RTCPRE[4:0]				
rw		rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRE2[2:0]			PPRE1[2:0]			Reserved	HPRE[3:0]				SWS1	SWS0	SW1	SW0	
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	r	r	rw	rw	

Bits 31:30 **MCO2[1:0]: Microcontroller clock output 2**

Set and cleared by software. Clock source selection may generate glitches on MCO2. It is highly recommended to configure these bits only after reset before enabling the external oscillators and the PLLs.

- 00: System clock (SYSCLK) selected
- 01: PLLI2S clock selected
- 10: HSE oscillator clock selected
- 11: PLL clock selected

Bits 27:29 **MCO2PRE: MCO2 prescaler**

Set and cleared by software to configure the prescaler of the MCO2. Modification of this prescaler may generate glitches on MCO2. It is highly recommended to change this prescaler only after reset before enabling the external oscillators and the PLLs.

- 0xx: no division
- 100: division by 2
- 101: division by 3
- 110: division by 4
- 111: division by 5

Bits 24:26 **MCO1PRE: MCO1 prescaler**

Set and cleared by software to configure the prescaler of the MCO1. Modification of this prescaler may generate glitches on MCO1. It is highly recommended to change this prescaler only after reset before enabling the external oscillators and the PLL.

- 0xx: no division
- 100: division by 2
- 101: division by 3
- 110: division by 4
- 111: division by 5

Bit 23 **I2SSRC: I2S clock selection**

Set and cleared by software. This bit allows to select the I2S clock source between the PLLI2S clock and the external clock. It is highly recommended to change this bit only after reset and before enabling the I2S module.

- 0: PLLI2S clock used as I2S clock source
- 1: External clock mapped on the I2S_CKIN pin used as I2S clock source

Bits 22:21 MCO1: Microcontroller clock output 1

Set and cleared by software. Clock source selection may generate glitches on MCO1. It is highly recommended to configure these bits only after reset before enabling the external oscillators and PLL.

- 00: HSI clock selected
- 01: LSE oscillator selected
- 10: HSE oscillator clock selected
- 11: PLL clock selected

Bits 20:16 RTCPRE: HSE division factor for RTC clock

Set and cleared by software to divide the HSE clock input clock to generate a 1 MHz clock for RTC.

Caution: The software has to set these bits correctly to ensure that the clock supplied to the RTC is 1 MHz. These bits must be configured if needed before selecting the RTC clock source.

- 00000: no clock
- 00001: no clock
- 00010: HSE/2
- 00011: HSE/3
- 00100: HSE/4
- ...
- 11110: HSE/30
- 11111: HSE/31

Bits 15:13 PPRE2: APB high-speed prescaler (APB2)

Set and cleared by software to control APB high-speed clock division factor.

Caution: The software has to set these bits correctly not to exceed 84 MHz on this domain. The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after PPRE2 write.

- 0xx: AHB clock not divided
- 100: AHB clock divided by 2
- 101: AHB clock divided by 4
- 110: AHB clock divided by 8
- 111: AHB clock divided by 16

Bits 12:10 PPRE1: APB Low speed prescaler (APB1)

Set and cleared by software to control APB low-speed clock division factor.

Caution: The software has to set these bits correctly not to exceed 42 MHz on this domain. The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after PPRE1 write.

- 0xx: AHB clock not divided
- 100: AHB clock divided by 2
- 101: AHB clock divided by 4
- 110: AHB clock divided by 8
- 111: AHB clock divided by 16

Bits 9:8 Reserved, must be kept at reset value.

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

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

Caution: The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after HPRE write.

Caution: The AHB clock frequency must be at least 25 MHz when the Ethernet is used.

0xxx: system clock not divided

1000: system clock divided by 2

1001: system clock divided by 4

1010: system clock divided by 8

1011: system clock divided by 16

1100: system clock divided by 64

1101: system clock divided by 128

1110: system clock divided by 256

1111: system clock divided by 512

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

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

00: HSI oscillator used as the system clock

01: HSE oscillator used as the system clock

10: PLL used as the system clock

11: not applicable

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

Set and cleared by software to select the system clock source.

Set by hardware to force the HSI selection when leaving the Stop or Standby mode or in case of failure of the HSE oscillator used directly or indirectly as the system clock.

00: HSI oscillator selected as system clock

01: HSE oscillator selected as system clock

10: PLL selected as system clock

11: not allowed

5.3.4 RCC clock interrupt register (RCC_CIR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CSSC	Reser ved	PLL2S RDYC	PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
15	14	13	12	11	10	9	8	7		W	W	W	W	W	W
Reserved		PLL2S RDYIE	PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	Reser ved	PLL2S RDYF	PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
		rw	rw	rw	rw	rw	rw	r		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 Reserved, must be kept at reset value.

Bit 21 **PLL2SRDYC:** PLLI2S ready interrupt clear

This bit is set by software to clear the PLLI2SRDYF flag.

0: No effect

1: PLLI2SRDYF cleared

Bit 20 **PLLRDYC:** Main PLL(PLL) ready interrupt clear

This bit is set by software to clear the PLLRDYF flag.

0: No effect

1: PLLRDYF cleared

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

This bit is set by software to clear the HSERDYF flag.

0: No effect

1: HSERDYF cleared

Bit 18 **HSIRDYC:** HSI ready interrupt clear

This bit is set by software to clear the HSIRDYF flag.

0: No effect

1: HSIRDYF cleared

Bit 17 **LSERDYC:** LSE ready interrupt clear

This bit is set by software to clear the LSERDYF flag.

0: No effect

1: LSERDYF cleared

Bit 16 **LSIRDYC:** LSI ready interrupt clear

This bit is set by software to clear the LSIRDYF flag.

0: No effect

1: LSIRDYF cleared

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

Bit 13 **PLL2SRDYIE:** PLLI2S ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by PLLI2S lock.

0: PLLI2S lock interrupt disabled

1: PLLI2S lock interrupt enabled

Bit 12 **PLLRDYIE:** Main PLL (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 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 Reserved, must be kept at reset value.
- Bit 5 **PLL12SRDYF:** PLLI2S ready interrupt flag
Set by hardware when the PLLI2S locks and PLLI2SRDYDIE is set.
Cleared by software setting the PLLRI2SDYFC bit.
0: No clock ready interrupt caused by PLLI2S lock
1: Clock ready interrupt caused by PLLI2S lock
- Bit 4 **PLLRDYF:** Main PLL (PLL) ready interrupt flag
Set by hardware when PLL locks and PLLRDYDIE is set.
Cleared by software setting the PLLRDYFC 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 External High Speed clock becomes stable and HSERDYDIE is set.
Cleared by software setting the HSERDYFC bit.
0: No clock ready interrupt caused by the HSE oscillator
1: Clock ready interrupt caused by the HSE oscillator
- Bit 2 **HSIRDYF:** HSI ready interrupt flag
Set by hardware when the Internal High Speed clock becomes stable and HSIRDYDIE is set.
Cleared by software setting the HSIRDYFC 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 External Low Speed clock becomes stable and LSERDYDIE is set.
Cleared by software setting the LSERDYFC 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 internal low speed clock becomes stable and LSIRDYDIE is set.
Cleared by software setting the LSIRDYC bit.

0: No clock ready interrupt caused by the LSI oscillator
1: Clock ready interrupt caused by the LSI oscillator

5.3.5 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

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
Reserved	OTGHS RST	Reserved				ETHMAC RST	Reserved		DMA2 RST	DMA1 RST	Reserved				
									rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CRCRS T	Reserved				GPIOI RST	GPIOH RST	GPIOGG RST	GPIOF RST	GPIOE RST	GPIOD RST	GPIOC RST	GPIOB RST	GPIOA RST	
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 29 **OTGHSRST:** USB OTG HS module reset

Set and cleared by software.
0: does not reset the USB OTG HS module
1: resets the USB OTG HS module

Bits 28:26 Reserved, must be kept at reset value.

Bit 25 **ETHMACRST:** Ethernet MAC reset

Set and cleared by software.
0: does not reset Ethernet MAC
1: resets Ethernet MAC

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **DMA2RST:** DMA2 reset

Set and cleared by software.
0: does not reset DMA2
1: resets DMA2

Bit 21 **DMA1RST:** DMA2 reset

Set and cleared by software.
0: does not reset DMA2
1: resets DMA2

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **CRCRST:** CRC reset

Set and cleared by software.
0: does not reset CRC
1: resets CRC

Bits 11:9 Reserved, must be kept at reset value.

- Bit 8 **GPIOIRST:** IO port I reset
Set and cleared by software.
0: does not reset IO port I
1: resets IO port I
- Bit 7 **GPIOHRST:** IO port H reset
Set and cleared by software.
0: does not reset IO port H
1: resets IO port H
- Bits 6 **GPIOGRST:** IO port G reset
Set and cleared by software.
0: does not reset IO port G
1: resets IO port G
- Bit 5 **GPIOFRST:** IO port F reset
Set and cleared by software.
0: does not reset IO port F
1: resets IO port F
- Bit 4 **GPIOERST:** IO port E reset
Set and cleared by software.
0: does not reset IO port E
1: resets IO port E
- Bit 3 **GPIODRST:** IO port D reset
Set and cleared by software.
0: does not reset IO port D
1: resets IO port D
- Bit 2 **GPIOCRST:** IO port C reset
Set and cleared by software.
0: does not reset IO port C
1: resets IO port C
- Bit 1 **GPIOBRST:** IO port B reset
Set and cleared by software.
0: does not reset IO port B
1: resets IO port B
- Bit 0 **GPIOARST:** IO port A reset
Set and cleared by software.
0: does not reset IO port A
1: resets IO port A

5.3.6 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x14

Reset value: 0x0000 0000

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

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

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

Bit 7 **OTGFSRST:** USB OTG FS module reset

Set and cleared by software.

0: does not reset the USB OTG FS module

1: resets the USB OTG FS module

Bit 6 **RNGRST:** Random number generator module reset

Set and cleared by software.

0: does not reset the random number generator module

1: resets the random number generator module

Bit 5 **HASHRST:** Hash module reset

Set and cleared by software.

0: does not reset the HASH module

1: resets the HASH module

Bit 4 **CRYPRST:** Cryptographic module reset

Set and cleared by software.

0: does not reset the cryptographic module

1: resets the cryptographic module

Bit 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMIRST:** Camera interface reset

Set and cleared by software.

0: does not reset the Camera interface

1: resets the Camera interface

5.3.7 RCC AHB3 peripheral reset register (RCC_AHB3RSTR)

Address offset: 0x18

Reset value: 0x0000 0000

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

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

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

Bit 0 **FSMCRST**: Flexible static memory controller module reset

Set and cleared by software.

0: does not reset the FSMC module

1: resets the FSMC module

5.3.8 RCC APB1 peripheral reset register (RCC_APB1RSTR)

Address offset: 0x20

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DACRST	PWR RST	Reserved	CAN2 RST	CAN1 RST	Reserved	I2C3 RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	UART3 RST	UART2 RST	Reserved
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	Reserved		WWDG RST	Reserved		TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 29 **DACRST**: DAC reset

Set and cleared by software.

0: does not reset the DAC interface

1: resets the DAC interface

Bit 28 **PWRRST**: Power interface reset

Set and cleared by software.

0: does not reset the power interface

1: resets the power interface

Bit 27 Reserved, must be kept at reset value.

- Bit 26 **CAN2RST:** CAN2 reset
Set and cleared by software.
0: does not reset CAN2
1: resets CAN2
- Bit 25 **CAN1RST:** CAN1 reset
Set and cleared by software.
0: does not reset CAN1
1: resets CAN1
- Bit 24 Reserved, must be kept at reset value.
- Bit 23 **I2C3RST:** I2C3 reset
Set and cleared by software.
0: does not reset I2C3
1: resets I2C3
- Bit 22 **I2C2RST:** I2C 2 reset
Set and cleared by software.
0: does not reset I2C2
1: resets I2C 2
- Bit 21 **I2C1RST:** I2C 1 reset
Set and cleared by software.
0: does not reset I2C1
1: resets I2C1
- Bit 20 **UART5RST:** USART 5 reset
Set and cleared by software.
0: does not reset USART5
1: resets USART5
- Bit 19 **UART4RST:** USART 4 reset
Set and cleared by software.
0: does not reset USART4
1: resets USART4
- Bit 18 **USART3RST:** USART 3 reset
Set and cleared by software.
0: does not reset USART3
1: resets USART3
- Bit 17 **USART2RST:** USART 2 reset
Set and cleared by software.
0: does not reset USART2
1: resets USART2
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **SPI3RST:** SPI 3 reset
Set and cleared by software.
0: does not reset SPI3
1: resets SPI3
- Bit 14 **SPI2RST:** SPI 2 reset
Set and cleared by software.
0: does not reset SPI2
1: resets SPI2

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGRST:** Window watchdog reset
Set and cleared by software.
0: does not reset the window watchdog
1: resets the window watchdog

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **TIM14RST:** TIM14 reset
Set and cleared by software.
0: does not reset TIM14
1: resets TIM14

Bit 7 **TIM13RST:** TIM13 reset
Set and cleared by software.
0: does not reset TIM13
1: resets TIM13

Bit 6 **TIM12RST:** TIM12 reset
Set and cleared by software.
0: does not reset TIM12
1: resets TIM12

Bit 5 **TIM7RST:** TIM7 reset
Set and cleared by software.
0: does not reset TIM7
1: resets TIM7

Bit 4 **TIM6RST:** TIM6 reset
Set and cleared by software.
0: does not reset TIM6
1: resets TIM6

Bit 3 **TIM5RST:** TIM5 reset
Set and cleared by software.
0: does not reset TIM5
1: resets TIM5

Bit 2 **TIM4RST:** TIM4 reset
Set and cleared by software.
0: does not reset TIM4
1: resets TIM4

Bit 1 **TIM3RST:** TIM3 reset
Set and cleared by software.
0: does not reset TIM3
1: resets TIM3

Bit 0 **TIM2RST:** TIM2 reset
Set and cleared by software.
0: does not reset TIM2
1: resets TIM2

5.3.9 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x24

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved														TIM11 RST	TIM10 RST	TIM9 RST
														rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	SYSCFG RST	Reser- ved	SPI1 RST	SDIO RST	Reserved	ADC RST	Reserved	USART6 RST	USART1 RST	Reserved	Reserved	TIM8 RST	TIM1 RST	rw	rw	rw

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

Bit 18 **TIM11RST:** TIM11 reset

Set and cleared by software.

0: does not reset TIM11
1: resets TIM14

Bit 17 **TIM10RST:** TIM10 reset

Set and cleared by software.

0: does not reset TIM10
1: resets TIM10

Bit 16 **TIM9RST:** TIM9 reset

Set and cleared by software.

0: does not reset TIM9
1: resets TIM9

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYSCFGRST:** System configuration controller reset

Set and cleared by software.

0: does not reset the System configuration controller
1: resets the System configuration controller

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1RST:** SPI 1 reset

Set and cleared by software.

0: does not reset SPI1
1: resets SPI1

Bit 11 **SDIORST:** SDIO reset

Set and cleared by software.

0: does not reset the SDIO module
1: resets the SDIO module

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **ADCRST:** ADC interface reset (common to all ADCs)

Set and cleared by software.

0: does not reset the ADC interface
1: resets the ADC interface

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

Bit 5 **USART6RST:** USART6 reset

Set and cleared by software.
0: does not reset USART6
1: resets USART6

Bit 4 **USART1RST:** USART1 reset

Set and cleared by software.
0: does not reset USART1
1: resets USART1

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

Bit 1 **TIM8RST:** TIM8 reset

Set and cleared by software.
0: does not reset TIM8
1: resets TIM8

Bit 0 **TIM1RST:** TIM1 reset

Set and cleared by software.
0: does not reset TIM1
1: resets TIM1

5.3.10 RCC AHB1 peripheral clock register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0010 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
Reser- ved	OTGHS ULPIEN	OTGHS EN	ETHMA CPTPE N	ETHMA CRXEN	ETHMA CTXEN	ETHMA CEN	Reserved	DMA2EN	DMA1EN	CCMDATA RAMEN	Res.	BKPSR AMEN	Reserved	Reserved	Reserved	Reserved
	rw	rw	rw	rw	rw	rw		rw	rw			rw				
	15	14	13	12	11	10		8	7	6	5	4	3	2	1	0
Reserved		CRCEN	Reserved			GPIOE N	GPIOH EN	GPIOGE N	GPIOFE N	GPIOEEN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN		
		rw				rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31 Reserved, must be kept at reset value.

Bit 30 **OTGHSULPIEN:** USB OTG HSULPI clock enable

Set and cleared by software.

- 0: USB OTG HS ULPI clock disabled
- 1: USB OTG HS ULPI clock enabled

Bit 29 **OTGHSEN:** USB OTG HS clock enable

Set and cleared by software.

- 0: USB OTG HS clock disabled
- 1: USB OTG HS clock enabled

Bit 28 **ETHMACPTPEN:** Ethernet PTP clock enable

Set and cleared by software.

- 0: Ethernet PTP clock disabled
- 1: Ethernet PTP clock enabled

Bit 27 **ETHMACRXEN:** Ethernet Reception clock enable

Set and cleared by software.

- 0: Ethernet Reception clock disabled
- 1: Ethernet Reception clock enabled

Bit 26 **ETHMACTXEN:** Ethernet Transmission clock enable

Set and cleared by software.

- 0: Ethernet Transmission clock disabled
- 1: Ethernet Transmission clock enabled

Bit 25 **ETHMACEN:** Ethernet MAC clock enable

Set and cleared by software.

- 0: Ethernet MAC clock disabled
- 1: Ethernet MAC clock enabled

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **DMA2EN:** DMA2 clock enable

Set and cleared by software.

- 0: DMA2 clock disabled
- 1: DMA2 clock enabled

- Bit 21 **DMA1EN:** DMA1 clock enable
Set and cleared by software.
0: DMA1 clock disabled
1: DMA1 clock enabled
- Bit 20 **CCMDATARAMEN:** CCM data RAM clock enable
Set and cleared by software.
0: CCM data RAM clock disabled
1: CCM data RAM clock enabled
- Bits 19 Reserved, must be kept at reset value.
- Bit 18 **BKPSRAMEN:** Backup SRAM interface clock enable
Set and cleared by software.
0: Backup SRAM interface clock disabled
1: Backup SRAM interface clock enabled
- Bits 17:13 Reserved, must be kept at reset value.
- Bit 12 **CRCEN:** CRC clock enable
Set and cleared by software.
0: CRC clock disabled
1: CRC clock enabled
- Bits 11:9 Reserved, must be kept at reset value.
- Bit 8 **GPIOIEN:** IO port I clock enable
Set and cleared by software.
0: IO port I clock disabled
1: IO port I clock enabled
- Bit 7 **GPIOHEN:** IO port H clock enable
Set and cleared by software.
0: IO port H clock disabled
1: IO port H clock enabled
- Bit 6 **GPIOGEN:** IO port G clock enable
Set and cleared by software.
0: IO port G clock disabled
1: IO port G clock enabled
- Bit 5 **GPIOFEN:** IO port F clock enable
Set and cleared by software.
0: IO port F clock disabled
1: IO port F clock enabled
- Bit 4 **GPIOEEN:** IO port E clock enable
Set and cleared by software.
0: IO port E clock disabled
1: IO port E clock enabled
- Bit 3 **GIPODEN:** IO port D clock enable
Set and cleared by software.
0: IO port D clock disabled
1: IO port D clock enabled

Bit 2 **GPIOCEN**: IO port C clock enable

Set and cleared by software.

0: IO port C clock disabled

1: IO port C clock enabled

Bit 1 **GPIOBEN**: IO port B clock enable

Set and cleared by software.

0: IO port B clock disabled

1: IO port B clock enabled

Bit 0 **GPIOAEN**: IO port A clock enable

Set and cleared by software.

0: IO port A clock disabled

1: IO port A clock enabled

5.3.11 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x34

Reset value: 0x0000 0000

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

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

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

Bit 7 **OTGFSEN**: USB OTG FS clock enable

Set and cleared by software.

0: USB OTG FS clock disabled

1: USB OTG FS clock enabled

Bit 6 **RNGEN**: Random number generator clock enable

Set and cleared by software.

0: Random number generator clock disabled

1: Random number generator clock enabled

Bit 5 **HASHEN**: Hash modules clock enable

Set and cleared by software.

0: Hash modules clock disabled

1: Hash modules clock enabled

Bit 4 **CRYPEN**: Cryptographic modules clock enable

Set and cleared by software.

0: cryptographic module clock disabled

1: cryptographic module clock enabled

Bit 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMIEN**: Camera interface enable

Set and cleared by software.

0: Camera interface clock disabled

1: Camera interface clock enabled

5.3.12 RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)

Address offset: 0x38

Reset value: 0x0000 0000

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

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

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

Bit 0 **FSMCEN**: Flexible static memory controller module clock enable

Set and cleared by software.

0: FSMC module clock disabled

1: FSMC module clock enabled

5.3.13 RCC APB1 peripheral clock enable register (RCC_APB1ENR)

Address offset: 0x40

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC EN	PWR EN	Reser- ved	CAN2 EN	CAN1 EN	Reser- ved	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	Reser- ved
		rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	
		rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWDG EN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 29 **DACEN:** DAC interface clock enable

Set and cleared by software.

0: DAC interface clock disabled

1: DAC interface clock enable

Bit 28 **PWREN:** Power interface clock enable

Set and cleared by software.

0: Power interface clock disabled

1: Power interface clock enable

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CAN2EN:** CAN 2 clock enable

Set and cleared by software.

0: CAN 2 clock disabled

1: CAN 2 clock enabled

Bit 25 **CAN1EN:** CAN 1 clock enable

Set and cleared by software.

0: CAN 1 clock disabled

1: CAN 1 clock enabled

Bit 24 Reserved, must be kept at reset value.

Bit 23 **I2C3EN:** I2C3 clock enable

Set and cleared by software.

0: I2C3 clock disabled

1: I2C3 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

Set and cleared by software.

0: UART5 clock disabled

1: UART5 clock enabled

Bit 19 **UART4EN:** UART4 clock enable

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:** USART 2 clock enable
Set and cleared by software.
0: USART2 clock disabled
1: USART2 clock enabled
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **SPI3EN:** SPI3 clock enable
Set and cleared by software.
0: SPI3 clock disabled
1: SPI3 clock enabled
- Bit 14 **SPI2EN:** SPI2 clock enable
Set and cleared by software.
0: SPI2 clock disabled
1: SPI2 clock enabled
- Bits 13:12 Reserved, must be kept at reset value.
- Bit 11 **WWDGEN:** Window watchdog clock enable
Set and cleared by software.
0: Window watchdog clock disabled
1: Window watchdog clock enabled
- Bit 10:9 Reserved, must be kept at reset value.
- Bit 8 **TIM14EN:** TIM14 clock enable
Set and cleared by software.
0: TIM14 clock disabled
1: TIM14 clock enabled
- Bit 7 **TIM13EN:** TIM13 clock enable
Set and cleared by software.
0: TIM13 clock disabled
1: TIM13 clock enabled
- Bit 6 **TIM12EN:** TIM12 clock enable
Set and cleared by software.
0: TIM12 clock disabled
1: TIM12 clock enabled
- Bit 5 **TIM7EN:** TIM7 clock enable
Set and cleared by software.
0: TIM7 clock disabled
1: TIM7 clock enabled
- Bit 4 **TIM6EN:** TIM6 clock enable
Set and cleared by software.
0: TIM6 clock disabled
1: TIM6 clock enabled
- Bit 3 **TIM5EN:** TIM5 clock enable
Set and cleared by software.
0: TIM5 clock disabled
1: TIM5 clock enabled

Bit 2 **TIM4EN:** TIM4 clock enable

Set and cleared by software.

0: TIM4 clock disabled

1: TIM4 clock enabled

Bit 1 **TIM3EN:** TIM3 clock enable

Set and cleared by software.

0: TIM3 clock disabled

1: TIM3 clock enabled

Bit 0 **TIM2EN:** TIM2 clock enable

Set and cleared by software.

0: TIM2 clock disabled

1: TIM2 clock enabled

5.3.14 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x44

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
												TIM11 EN	TIM10 EN	TIM9 EN	
												rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser- ved	SYSCF G EN	Reser- ved	SPI1 EN	SDIO EN	ADC3 EN	ADC2 EN	ADC1 EN	Reserved	Reserved	USART6 EN	USART1 EN	Reserved	TIM8 EN	TIM1 EN	
	rw		rw	rw	rw	rw	rw			rw	rw		rw	rw	

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

Bit 18 **TIM11EN:** TIM11 clock enable

Set and cleared by software.

- 0: TIM11 clock disabled
- 1: TIM11 clock enabled

Bit 17 **TIM10EN:** TIM10 clock enable

Set and cleared by software.

- 0: TIM10 clock disabled
- 1: TIM10 clock enabled

Bit 16 **TIM9EN:** TIM9 clock enable

Set and cleared by software.

- 0: TIM9 clock disabled
- 1: TIM9 clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYSCFGGEN:** System configuration controller clock enable

Set and cleared by software.

- 0: System configuration controller clock disabled
- 1: System configuration controller clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN:** SPI1 clock enable

Set and cleared by software.

- 0: SPI1 clock disabled
- 1: SPI1 clock enabled

Bit 11 **SDIOEN:** SDIO clock enable

Set and cleared by software.

- 0: SDIO module clock disabled
- 1: SDIO module clock enabled

Bit 10 **ADC3EN:** ADC3 clock enable

Set and cleared by software.

- 0: ADC3 clock disabled
- 1: ADC3 clock disabled

Bit 9 **ADC2EN:** ADC2 clock enable
Set and cleared by software.
0: ADC2 clock disabled
1: ADC2 clock disabled

Bit 8 **ADC1EN:** ADC1 clock enable
Set and cleared by software.
0: ADC1 clock disabled
1: ADC1 clock disabled

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

Bit 5 **USART6EN:** USART6 clock enable
Set and cleared by software.
0: USART6 clock disabled
1: USART6 clock enabled

Bit 4 **USART1EN:** USART1 clock enable
Set and cleared by software.
0: USART1 clock disabled
1: USART1 clock enabled

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

Bit 1 **TIM8EN:** TIM8 clock enable
Set and cleared by software.
0: TIM8 clock disabled
1: TIM8 clock enabled

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

5.3.15 RCC AHB1 peripheral clock enable in low power mode register (RCC_AHB1LPENR)

Address offset: 0x50

Reset value: 0x7E67 91FF

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	
Reser- ved	OTGHS ULPILPEN	OTGHS LPEN	ETHPTP LPEN	ETHRX LPEN	ETHTX LPEN	ETHMAC LPEN	Reserved		DMA2 LPEN	DMA1 LPEN	Reserved		BKPSRA M LPEN	SRAM2 LPEN	SRAM1 LPEN	
	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
FLITF LPEN	Reserved	CRC LPEN	Reserved			GPIOI LPEN	GPIOH LPEN	GPIOGG LPEN	GPIOF LPEN	GPIOE LPEN	GPIOD LPEN	GPIOC LPEN	GPIOB LPEN	GPIOA LPEN		
rw		rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **OTGHSULPILPEN:** USB OTG HS ULPI clock enable during Sleep mode

Set and cleared by software.

0: USB OTG HS ULPI clock disabled during Sleep mode

1: USB OTG HS ULPI clock enabled during Sleep mode

Bit 29 **OTGHSLPEN:** USB OTG HS clock enable during Sleep mode

Set and cleared by software.

0: USB OTG HS clock disabled during Sleep mode

1: USB OTG HS clock enabled during Sleep mode

Bit 28 **ETHMACPTPLPEN:** Ethernet PTP clock enable during Sleep mode

Set and cleared by software.

0: Ethernet PTP clock disabled during Sleep mode

1: Ethernet PTP clock enabled during Sleep mode

Bit 27 **ETHMACRXLPEN:** Ethernet reception clock enable during Sleep mode

Set and cleared by software.

0: Ethernet reception clock disabled during Sleep mode

1: Ethernet reception clock enabled during Sleep mode

Bit 26 **ETHMACTXLPEN:** Ethernet transmission clock enable during Sleep mode

Set and cleared by software.

0: Ethernet transmission clock disabled during sleep mode

1: Ethernet transmission clock enabled during sleep mode

Bit 25 **ETHMACLPEN:** Ethernet MAC clock enable during Sleep mode

Set and cleared by software.

0: Ethernet MAC clock disabled during Sleep mode

1: Ethernet MAC clock enabled during Sleep mode

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **DMA2LPEN:** DMA2 clock enable during Sleep mode

Set and cleared by software.

0: DMA2 clock disabled during Sleep mode

1: DMA2 clock enabled during Sleep mode

- Bit 21 **DMA1LPEN:** DMA1 clock enable during Sleep mode
Set and cleared by software.
0: DMA1 clock disabled during Sleep mode
1: DMA1 clock enabled during Sleep mode
- Bits 20:19 Reserved, must be kept at reset value.
- Bit 18 **BKPSRAMLPEN:** Backup SRAM interface clock enable during Sleep mode
Set and cleared by software.
0: Backup SRAM interface clock disabled during Sleep mode
1: Backup SRAM interface clock enabled during Sleep mode
- Bit 17 **SRAM2LPEN:** SRAM 2 interface clock enable during Sleep mode
Set and cleared by software.
0: SRAM 2 interface clock disabled during Sleep mode
1: SRAM 2 interface clock enabled during Sleep mode
- Bit 16 **SRAM1LPEN:** SRAM 1interface clock enable during Sleep mode
Set and cleared by software.
0: SRAM 1 interface clock disabled during Sleep mode
1: SRAM 1 interface clock enabled during Sleep mode
- Bit 15 **FLITFLPEN:** Flash interface clock enable during Sleep mode
Set and cleared by software.
0: Flash interface clock disabled during Sleep mode
1: Flash interface clock enabled during Sleep mode
- Bits 14:13 Reserved, must be kept at reset value.
- Bit 12 **CRCLPEN:** CRC clock enable during Sleep mode
Set and cleared by software.
0: CRC clock disabled during Sleep mode
1: CRC clock enabled during Sleep mode
- Bits 11:9 Reserved, must be kept at reset value.
- Bit 8 **GPIOILPEN:** IO port I clock enable during Sleep mode
Set and cleared by software.
0: IO port I clock disabled during Sleep mode
1: IO port I clock enabled during Sleep mode
- Bit 7 **GPIOHLPEN:** IO port H clock enable during Sleep mode
Set and cleared by software.
0: IO port H clock disabled during Sleep mode
1: IO port H clock enabled during Sleep mode
- Bits 6 **GPIOGLPEN:** IO port G clock enable during Sleep mode
Set and cleared by software.
0: IO port G clock disabled during Sleep mode
1: IO port G clock enabled during Sleep mode
- Bit 5 **GPIOFLPEN:** IO port F clock enable during Sleep mode
Set and cleared by software.
0: IO port F clock disabled during Sleep mode
1: IO port F clock enabled during Sleep mode

- Bit 4 **GPIOELPEN:** IO port E clock enable during Sleep mode
Set and cleared by software.
0: IO port E clock disabled during Sleep mode
1: IO port E clock enabled during Sleep mode
- Bit 3 **GPIODLPEN:** IO port D clock enable during Sleep mode
Set and cleared by software.
0: IO port D clock disabled during Sleep mode
1: IO port D clock enabled during Sleep mode
- Bit 2 **GPIOCLPEN:** IO port C clock enable during Sleep mode
Set and cleared by software.
0: IO port C clock disabled during Sleep mode
1: IO port C clock enabled during Sleep mode
- Bit 1 **GPIOBLPEN:** IO port B clock enable during Sleep mode
Set and cleared by software.
0: IO port B clock disabled during Sleep mode
1: IO port B clock enabled during Sleep mode
- Bit 0 **GPIOALPEN:** IO port A clock enable during sleep mode
Set and cleared by software.
0: IO port A clock disabled during Sleep mode
1: IO port A clock enabled during Sleep mode

5.3.16 RCC AHB2 peripheral clock enable in low power mode register (RCC_AHB2LPENR)

Address offset: 0x54

Reset value: 0x0000 00F1

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

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

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

- Bit 7 **OTGFSLPEN:** USB OTG FS clock enable during Sleep mode
Set and cleared by software.
0: USB OTG FS clock disabled during Sleep mode
1: USB OTG FS clock enabled during Sleep mode

- Bit 6 **RNGLPEN:** Random number generator clock enable during Sleep mode
Set and cleared by software.
0: Random number generator clock disabled during Sleep mode
1: Random number generator clock enabled during Sleep mode

- Bit 5 **HASHPEN:** Hash modules clock enable during Sleep mode
Set and cleared by software.
0: Hash modules clock disabled during Sleep mode
1: Hash modules clock enabled during Sleep mode
- Bit 4 **CRYPLPEN:** Cryptography modules clock enable during Sleep mode
Set and cleared by software.
0: cryptography modules clock disabled during Sleep mode
1: cryptography modules clock enabled during Sleep mode
- Bit 3:1 Reserved, must be kept at reset value.
- Bit 0 **DCMILPEN:** Camera interface enable during Sleep mode
Set and cleared by software.
0: Camera interface clock disabled during Sleep mode
1: Camera interface clock enabled during Sleep mode

5.3.17 RCC AHB3 peripheral clock enable in low power mode register (RCC_AHB3LPENR)

Address offset: 0x58

Reset value: 0x0000 0001

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

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

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

- FSMCLPEN:** Flexible static memory controller module clock enable during Sleep mode
Set and cleared by software.
0: FSMC module clock disabled during Sleep mode
1: FSMC module clock enabled during Sleep mode

5.3.18 RCC APB1 peripheral clock enable in low power mode register (RCC_APB1LPENR)

Address offset: 0x60

Reset value: 0x36FE C9FF

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC LPEN	PWR LPEN	RESER VED	CAN2 LPEN	CAN1 LPEN	Reser- ved	I2C3 LPEN	I2C2 LPEN	I2C1 LPEN	UART5 LPEN	UART4 LPEN	USART3 LPEN	USART2 LPEN	Reser- ved
		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 LPEN	SPI2 LPEN	Reserved	WWDG LPEN	Reserved	TIM14 LPEN	TIM13 LPEN	TIM12 LPEN	TIM7 LPEN	TIM6 LPEN	TIM5 LPEN	TIM4 LPEN	TIM3 LPEN	TIM2 LPEN		
rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 29 **DACLPEN:** DAC interface clock enable during Sleep mode

Set and cleared by software.

0: DAC interface clock disabled during Sleep mode

1: DAC interface clock enabled during Sleep mode

Bit 28 **PWRLPEN:** Power interface clock enable during Sleep mode

Set and cleared by software.

0: Power interface clock disabled during Sleep mode

1: Power interface clock enabled during Sleep mode

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CAN2LPEN:** CAN 2 clock enable during Sleep mode

Set and cleared by software.

0: CAN 2 clock disabled during sleep mode

1: CAN 2 clock enabled during sleep mode

Bit 25 **CAN1LPEN:** CAN 1 clock enable during Sleep mode

Set and cleared by software.

0: CAN 1 clock disabled during Sleep mode

1: CAN 1 clock enabled during Sleep mode

Bit 24 Reserved, must be kept at reset value.

Bit 23 **I2C3LPEN:** I2C3 clock enable during Sleep mode

Set and cleared by software.

0: I2C3 clock disabled during Sleep mode

1: I2C3 clock enabled during Sleep mode

Bit 22 **I2C2LPEN:** I2C2 clock enable during Sleep mode

Set and cleared by software.

0: I2C2 clock disabled during Sleep mode

1: I2C2 clock enabled during Sleep mode

Bit 21 **I2C1LPEN:** I2C1 clock enable during Sleep mode

Set and cleared by software.

0: I2C1 clock disabled during Sleep mode

1: I2C1 clock enabled during Sleep mode

- Bit 20 **UART5LPEN:** UART5 clock enable during Sleep mode
Set and cleared by software.
0: UART5 clock disabled during Sleep mode
1: UART5 clock enabled during Sleep mode
- Bit 19 **UART4LPEN:** UART4 clock enable during Sleep mode
Set and cleared by software.
0: UART4 clock disabled during Sleep mode
1: UART4 clock enabled during Sleep mode
- Bit 18 **USART3LPEN:** USART3 clock enable during Sleep mode
Set and cleared by software.
0: USART3 clock disabled during Sleep mode
1: USART3 clock enabled during Sleep mode
- Bit 17 **USART2LPEN:** USART2 clock enable during Sleep mode
Set and cleared by software.
0: USART2 clock disabled during Sleep mode
1: USART2 clock enabled during Sleep mode
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **SPI3LPEN:** SPI3 clock enable during Sleep mode
Set and cleared by software.
0: SPI3 clock disabled during Sleep mode
1: SPI3 clock enabled during Sleep mode
- Bit 14 **SPI2LPEN:** SPI2 clock enable during Sleep mode
Set and cleared by software.
0: SPI2 clock disabled during Sleep mode
1: SPI2 clock enabled during Sleep mode
- Bits 13:12 Reserved, must be kept at reset value.
- Bit 11 **WWDGLPEN:** Window watchdog clock enable during Sleep mode
Set and cleared by software.
0: Window watchdog clock disabled during sleep mode
1: Window watchdog clock enabled during sleep mode
- Bits 10:9 Reserved, must be kept at reset value.
- Bit 8 **TIM14LPEN:** TIM14 clock enable during Sleep mode
Set and cleared by software.
0: TIM14 clock disabled during Sleep mode
1: TIM14 clock enabled during Sleep mode
- Bit 7 **TIM13LPEN:** TIM13 clock enable during Sleep mode
Set and cleared by software.
0: TIM13 clock disabled during Sleep mode
1: TIM13 clock enabled during Sleep mode
- Bit 6 **TIM12LPEN:** TIM12 clock enable during Sleep mode
Set and cleared by software.
0: TIM12 clock disabled during Sleep mode
1: TIM12 clock enabled during Sleep mode

Bit 5 **TIM7LPEN:** TIM7 clock enable during Sleep mode

Set and cleared by software.

0: TIM7 clock disabled during Sleep mode

1: TIM7 clock enabled during Sleep mode

Bit 4 **TIM6LPEN:** TIM6 clock enable during Sleep mode

Set and cleared by software.

0: TIM6 clock disabled during Sleep mode

1: TIM6 clock enabled during Sleep mode

Bit 3 **TIM5LPEN:** TIM5 clock enable during Sleep mode

Set and cleared by software.

0: TIM5 clock disabled during Sleep mode

1: TIM5 clock enabled during Sleep mode

Bit 2 **TIM4LPEN:** TIM4 clock enable during Sleep mode

Set and cleared by software.

0: TIM4 clock disabled during Sleep mode

1: TIM4 clock enabled during Sleep mode

Bit 1 **TIM3LPEN:** TIM3 clock enable during Sleep mode

Set and cleared by software.

0: TIM3 clock disabled during Sleep mode

1: TIM3 clock enabled during Sleep mode

Bit 0 **TIM2LPEN:** TIM2 clock enable during Sleep mode

Set and cleared by software.

0: TIM2 clock disabled during Sleep mode

1: TIM2 clock enabled during Sleep mode

5.3.19 RCC APB2 peripheral clock enabled in low power mode register (RCC_APB2LPENR)

Address offset: 0x64

Reset value: 0x0007 5F33

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved														TIM11 LPEN	TIM10 LPEN	TIM9 LPEN
										rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	SYSCFG LPEN	Reser- ved	SPI1 LPEN	SDIO LPEN	ADC3 LPEN	ADC2 LPEN	ADC1 LPEN	Reserved	USART6 LPEN	USART1 LPEN	Reserved	TIM8 LPEN	TIM1 LPEN			
	rw		rw	rw	rw	rw	rw		rw	rw		rw	rw			

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

Bit 18 **TIM11LPEN:** TIM11 clock enable during Sleep mode

Set and cleared by software.

0: TIM11 clock disabled during Sleep mode

1: TIM11 clock enabled during Sleep mode

Bit 17 **TIM10LPEN:** TIM10 clock enable during Sleep mode

Set and cleared by software.

0: TIM10 clock disabled during Sleep mode

1: TIM10 clock enabled during Sleep mode

Bit 16 **TIM9LPEN:** TIM9 clock enable during sleep mode

Set and cleared by software.

0: TIM9 clock disabled during Sleep mode

1: TIM9 clock enabled during Sleep mode

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SYSCFGLPEN:** System configuration controller clock enable during Sleep mode

Set and cleared by software.

0: System configuration controller clock disabled during Sleep mode

1: System configuration controller clock enabled during Sleep mode

Bits 13 Reserved, must be kept at reset value.

Bit 12 **SPI1LPEN:** SPI 1 clock enable during Sleep mode

Set and cleared by software.

0: SPI 1 clock disabled during Sleep mode

1: SPI 1 clock enabled during Sleep mode

Bit 11 **SDIOLPEN:** SDIO clock enable during Sleep mode

Set and cleared by software.

0: SDIO module clock disabled during Sleep mode

1: SDIO module clock enabled during Sleep mode

Bit 10 **ADC3LPEN:** ADC 3 clock enable during Sleep mode

Set and cleared by software.

0: ADC 3 clock disabled during Sleep mode

1: ADC 3 clock disabled during Sleep mode

Bit 9 **ADC2LPEN:** ADC2 clock enable during Sleep mode

Set and cleared by software.

0: ADC2 clock disabled during Sleep mode

1: ADC2 clock disabled during Sleep mode

Bit 8 **ADC1LPEN:** ADC1 clock enable during Sleep mode

Set and cleared by software.

0: ADC1 clock disabled during Sleep mode

1: ADC1 clock disabled during Sleep mode

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

Bit 5 **USART6LPEN:** USART6 clock enable during Sleep mode

Set and cleared by software.

0: USART6 clock disabled during Sleep mode

1: USART6 clock enabled during Sleep mode

Bit 4 **USART1LPEN:** USART1 clock enable during Sleep mode

Set and cleared by software.

0: USART1 clock disabled during Sleep mode

1: USART1 clock enabled during Sleep mode

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

Bit 1 **TIM8LPEN:** TIM8 clock enable during Sleep mode

Set and cleared by software.

0: TIM8 clock disabled during Sleep mode

1: TIM8 clock enabled during Sleep mode

Bit 0 **TIM1LPEN:** TIM1 clock enable during Sleep mode

Set and cleared by software.

0: TIM1 clock disabled during Sleep mode

1: TIM1 clock enabled during Sleep mode

5.3.20 RCC Backup domain control register (RCC_BDCR)

Address offset: 0x70

Reset value: 0x0000 0000, reset by Backup domain reset.

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

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

The LSEON, LSEBYP, RTCSEL and RTCEN bits in the [RCC Backup domain control register \(RCC_BDCR\)](#) are in the Backup domain. As a result, after Reset, these bits are write-protected and the DBP bit in the [Power control register \(PWR_CR\)](#) has to be set before these can be modified. Refer to [Section 4.1.2 on page 51](#) for further information. These bits are only reset after a Backup domain Reset (see [Section 5.1.3: Backup domain reset](#)). Any internal or external Reset will not have any effect on these bits.

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

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

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

Set and cleared by software.

0: Reset not activated

1: Resets the entire Backup domain

Note: The BKPSRAM is not affected by this reset, the only way of resetting the BKPSRAM is through the Flash interface when a protection level change from level 1 to level 0 is requested.

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 Backup domain is reset. The BDRST bit can be used to reset them.

00: No clock

01: LSE oscillator clock used as the RTC clock

10: LSI oscillator clock used as the RTC clock

11: HSE oscillator clock divided by a programmable prescaler (selection through the RTCPRE[4:0] bits in the RCC clock configuration register (RCC_CFGR)) used as the RTC clock

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

Bit 2 **LSEBYP**: External low-speed oscillator bypass

Set and cleared by software to bypass oscillator in debug mode. This bit can be written only when the LSE clock is disabled.

0: LSE oscillator not bypassed

1: LSE oscillator bypassed

Bit 1 **LSERDY**: External low-speed oscillator ready

Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles.

0: LSE clock not ready
1: LSE clock ready

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

Set and cleared by software.
0: LSE clock OFF
1: LSE clock ON

5.3.21 RCC clock control & status register (RCC_CSR)

Address offset: 0x74

Reset value: 0x0E00 0000, reset by system reset, except reset flags by power reset only.

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	BORRS TF	RMVF	Reserved							
rw	rw	rw	rw	rw	rw	rw	rw	15	14	13	12	11	10	9	8
Reserved															
															LSIRDY
															r
															rw

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

Set by hardware when a Low-power management reset occurs.

Cleared by writing to the RMVF bit.

0: No Low-power management reset occurred

1: Low-power management reset occurred

For further information on Low-power management reset, refer to [Low-power management reset](#).

Bit 30 **WWDGRSTF**: Window watchdog reset flag

Set by hardware when a window watchdog reset occurs.

Cleared by writing to the RMVF bit.

0: No window watchdog reset occurred

1: Window watchdog reset occurred

Bit 29 **IWDGRSTF**: Independent watchdog reset flag

Set by hardware when an independent watchdog reset from V_{DD} domain occurs.

Cleared by writing to the RMVF bit.

0: No watchdog reset occurred

1: Watchdog reset occurred

Bit 28 **SFTRSTF**: Software reset flag

Set by hardware when a software reset occurs.

Cleared by writing to the RMVF bit.

0: No software reset occurred

1: Software reset occurred

Bit 27 PORRSTF: POR/PDR reset flag

Set by hardware when a POR/PDR reset occurs.

Cleared by writing to the RMVF bit.

0: No POR/PDR reset occurred

1: POR/PDR reset occurred

Bit 26 PINRSTF: PIN reset flag

Set by hardware when a reset from the NRST pin occurs.

Cleared by writing to the RMVF bit.

0: No reset from NRST pin occurred

1: Reset from NRST pin occurred

Bit 25 BORRSTF: BOR reset flag

Cleared by software by writing the RMVF bit.

Set by hardware when a POR/PDR or BOR reset occurs.

0: No POR/PDR or BOR reset occurred

1: POR/PDR or BOR reset occurred

Bit 24 RMVF: Remove reset flag

Set by software to clear the reset flags.

0: No effect

1: Clear the reset flags

Bits 23:2 Reserved, must be kept at reset value.

Bit 1 LSIRDY: Internal low-speed oscillator ready

Set and cleared by hardware to indicate when the internal RC 40 kHz oscillator is stable.

After the LSION bit is cleared, LSIRDY goes low after 3 LSI clock cycles.

0: LSI RC oscillator not ready

1: LSI RC oscillator ready

Bit 0 LSION: Internal low-speed oscillator enable

Set and cleared by software.

0: LSI RC oscillator OFF

1: LSI RC oscillator ON

5.3.22 RCC spread spectrum clock generation register (RCC_SSCGR)

Address offset: 0x80

Reset value: 0x0000 0000

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

The spread spectrum clock generation is available only for the main PLL.

The RCC_SSCGR register must be written either before the main PLL is enabled or after the main PLL disabled.

Note: For full details about PLL spread spectrum clock generation (SSCG) characteristics, refer to the “Electrical characteristics” section in your device datasheet.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCG EN	SPR EAD SEL	Reserved	INCSTEP												
			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
INCSTEP				MODPER											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31 **SSCGEN:** Spread spectrum modulation enable

Set and cleared by software.

0: Spread spectrum modulation DISABLE. (To write after clearing CR[24]=PLLON bit)

1: Spread spectrum modulation ENABLE. (To write before setting CR[24]=PLLON bit)

Bit 30 **SPREADSEL:** Spread Select

Set and cleared by software.

To write before to set CR[24]=PLLON bit.

0: Center spread

1: Down spread

Bit 29:28 Reserved, must be kept at reset value.

Bit 27:13 **INCSTEP:** Incrementation step

Set and cleared by software. To write before setting CR[24]=PLLON bit.

Configuration input for modulation profile amplitude.

Bit 12:0 **MODPER:** Modulation period

Set and cleared by software. To write before setting CR[24]=PLLON bit.

Configuration input for modulation profile period.

5.3.23 RCC PLLI2S configuration register (RCC_PLLI2SCFGR)

Address offset: 0x84

Reset value: 0x2000 3000

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

This register is used to configure the PLLI2S clock outputs according to the formulas:

- $f_{(VCO\ clock)} = f_{(PLL\ clock\ input)} \times (PLL\ SN / PLLM)$
- $f_{(PLL\ I2S\ clock\ output)} = f_{(VCO\ clock)} / PLL\ SR$

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	PLLI2S R2	PLLI2S R1	PLLI2S R0	Reserved												
	rw	rw	rw													
Reserved	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PLLI2SN 8	PLLI2SN 7	PLLI2SN 6	PLLI2SN 5	PLLI2SN 4	PLLI2SN 3	PLLI2SN 2	PLLI2SN 1	PLLI2SN 0							Reserved
	rw	rw	rw	rw	rw	rw	rw									

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **PLLI2SR**: PLLI2S division factor for I2S clocks

Set and cleared by software to control the I2S clock frequency. These bits should be written only if the PLLI2S is disabled. The factor must be chosen in accordance with the prescaler values inside the I2S peripherals, to reach 0.3% error when using standard crystals and 0% error with audio crystals. For more information about I2S clock frequency and precision, refer to [Section 25.4.4: Clock generator](#) in the I2S chapter.

Caution: The I2Ss requires a frequency lower than or equal to 192 MHz to work correctly.

I2S clock frequency = VCO frequency / PLLR with $2 \leq PLLR \leq 7$

000: PLLR = 0, wrong configuration

001: PLLR = 1, wrong configuration

010: PLLR = 2

...

111: PLLR = 7

Bits 27:15 Reserved, must be kept at reset value.

Bits 14:6 **PLL2SN**: PLLI2S multiplication factor for VCO

Set and cleared by software to control the multiplication factor of the VCO. These bits can be written only when the PLLI2S is disabled. Only half-word and word accesses are allowed to write these bits.

Caution: The software has to set these bits correctly to ensure that the VCO output frequency is between 192 and 432 MHz.

VCO output frequency = VCO input frequency \times PLLI2SN with $192 \leq \text{PLLI2SN} \leq 432$

00000000: PLLI2SN = 0, wrong configuration

00000001: PLLI2SN = 1, wrong configuration

...

01100000: PLLI2SN = 192

01100001: PLLI2SN = 193

01100010: PLLI2SN = 194

...

110110000: PLLI2SN = 432

110110001: PLLI2SN = 433, wrong configuration

...

111111111: PLLI2SN = 511, wrong configuration

Bits 5:0 Reserved, must be kept at reset value.

5.3.24 RCC register map

Table 13 gives the register map and reset values.

Table 13. RCC register map and reset values

Table 13. RCC register map and reset values (continued)

Addr. offset	Register name		
0x40	RCC_APB1ENR	OTGHSLPEN	31
0x44	RCC_APB2ENR	OTGHSLPEN	30
0x48	Reserved	DACEN	29
0x4C	Reserved	PWREN	28
0x50	RCC_AHB1LENR	ETHMACPTPLPEN	27
0x54	RCC_AHB2LENR	ETHMACRXLPEN	26
0x58	RCC_AHB3LENR	ETHMACTXLPEN	25
0x5C	Reserved	ETHMACLPEN	24
0x60	RCC_APB1LENR	DMA2LPEN	23
0x64	RCC_APB2LENR	DMA1LPEN	22
0x68	Reserved	UART5EN	21
0x6C	Reserved	UART4EN	20
0x70	RCC_BDCR	UART3EN	19
0x74	RCC_CSR	UART2EN	18
0x78	Reserved	TIM10EN	17
0x7C	Reserved	TIM9EN	16
0x80	RCC_SSCGR	SPI3EN	15
0x84	RCC_PLLI2S CFGR	SPI2EN	14
0x88	PLL2SRx	SPI1EN	13
0x8C	Reserved	CRC1EN	12
0x90	Reserved	SDIOEN	11
0x94	RTCSEL1	ADC3EN	10
0x98	RTCSEL0	ADC2EN	9
0xA0	Reserved	ADC1EN	8
0xA4	Reserved	TIM13EN	7
0xA8	Reserved	TIM12EN	6
0xB0	USART6EN	TIM7EN	5
0xB4	USART5EN	TIM6EN	4
0xB8	USART4EN	TIM5EN	3
0xC0	USART3EN	TIM4EN	2
0xC4	USART2EN	TIM3EN	1
0xC8	USART1EN	TIM2EN	0

Refer to [Table 1 on page 50](#) for the register boundary addresses.

6 General-purpose I/Os (GPIO)

6.1 GPIO 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 register (GPIOx_AFRH and GPIOx_AFRL).

6.2 GPIO main features

- Up to 16 I/Os under control
- 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 I/O configuration
- Analog function
- Alternate function input/output selection registers (at most 16 AFs per I/O)
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

6.3 GPIO functional description

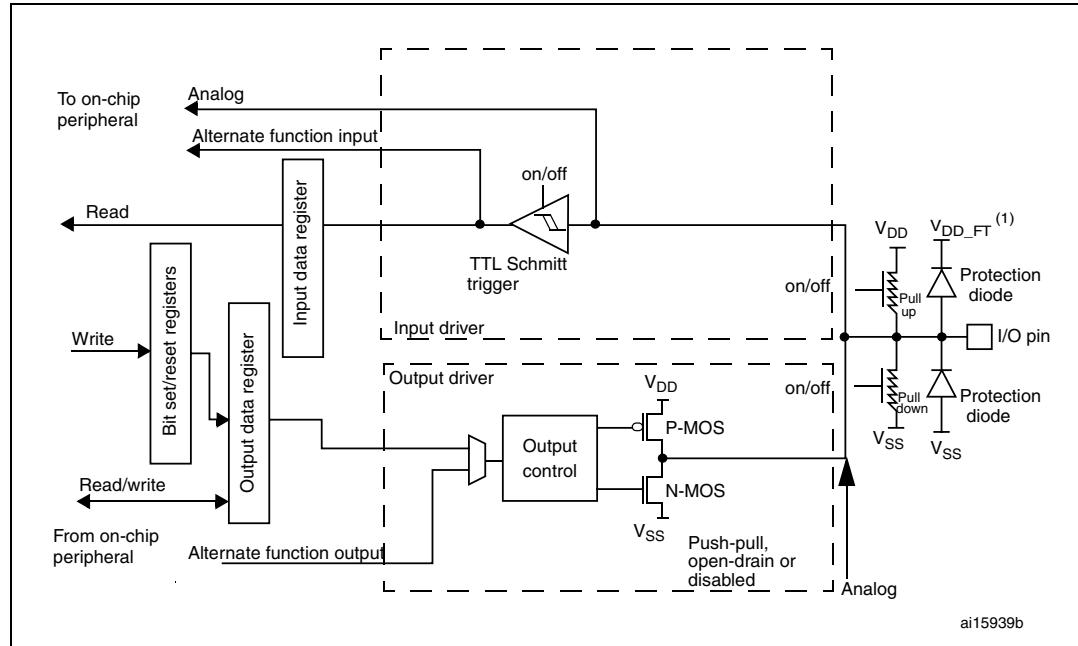
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

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

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR register is to allow atomic read/modify accesses to any of the GPIO registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 13 shows the basic structure of a 5 V tolerant I/O port bit. *Table 18* gives the possible port bit configurations.

Figure 13. Basic structure of a five-volt tolerant I/O port bit



ai15939b

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

Table 14. Port bit configuration table⁽¹⁾

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]	PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]	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)	

Table 14. Port bit configuration table⁽¹⁾ (continued)

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]	PUPDR(i) [1:0]		I/O configuration	
10	0	SPEED [B:A]	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
	x	x	x	0	1	Input
	x	x	x	1	0	Input
	x	x	x	1	1	Reserved (input floating)
11	x	x	x	0	0	Input/output
	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.

6.3.1 General-purpose I/O (GPIO)

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

The JTAG pins are in input pull-up/pull-down after reset:

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

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the N-MOS is activated when 0 is output).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB1 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.

6.3.2 I/O pin multiplexer and mapping

The STM32F40x and STM32F41x I/O pins are connected to onboard peripherals/modules through a multiplexer that allows only one peripheral's alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals sharing the same I/O pin.

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

- After reset all I/Os are connected to the system's alternate function 0 (AF0)
- The peripherals' alternate functions are mapped from AF1 to AF13
- CortexTM-M4F EVENTOUT is mapped on AF15

This structure is shown in [Figure 14](#) below.

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:

- **System function:** you have to connect the I/O to AF0 and configure it depending on the function used:
 - JTAG/SWD, after each device reset these pins are assigned as dedicated pins immediately usable by the debugger host (not controlled by the GPIO controller)
 - RTC_50Hz: this pin should be configured in Input floating mode
 - MCO1 and MCO2: these pins have to be configured in alternate function mode.

Note:

You can disable some or all of the JTAG/SWD pins and so release the associated pins for GPIO usage.

For more details please refer to [Section 5.2.10: Clock-out capability](#).

Table 15. Flexible SWJ-DP pin assignment

Available debug ports	SWJ I/O pin assigned				
	PA13 / JTMS/ SWDIO	PA14 / JTCK/ SWCLK	PA15 / JTDI	PB3 / JTDO	PB4/ NJTRST
Full SWJ (JTAG-DP + SW-DP) - Reset state	X	X	X	X	X
Full SWJ (JTAG-DP + SW-DP) but without NJTRST	X	X	X	X	
JTAG-DP Disabled and SW-DP Enabled	X	X			
JTAG-DP Disabled and SW-DP Disabled					Released

- **GPIO:** configure the desired I/O as output or input in the GPIOx_MODER register.

- **Peripheral's alternate function:**

For the ADC and DAC, configure the desired I/O as analog in the GPIOx_MODER register.

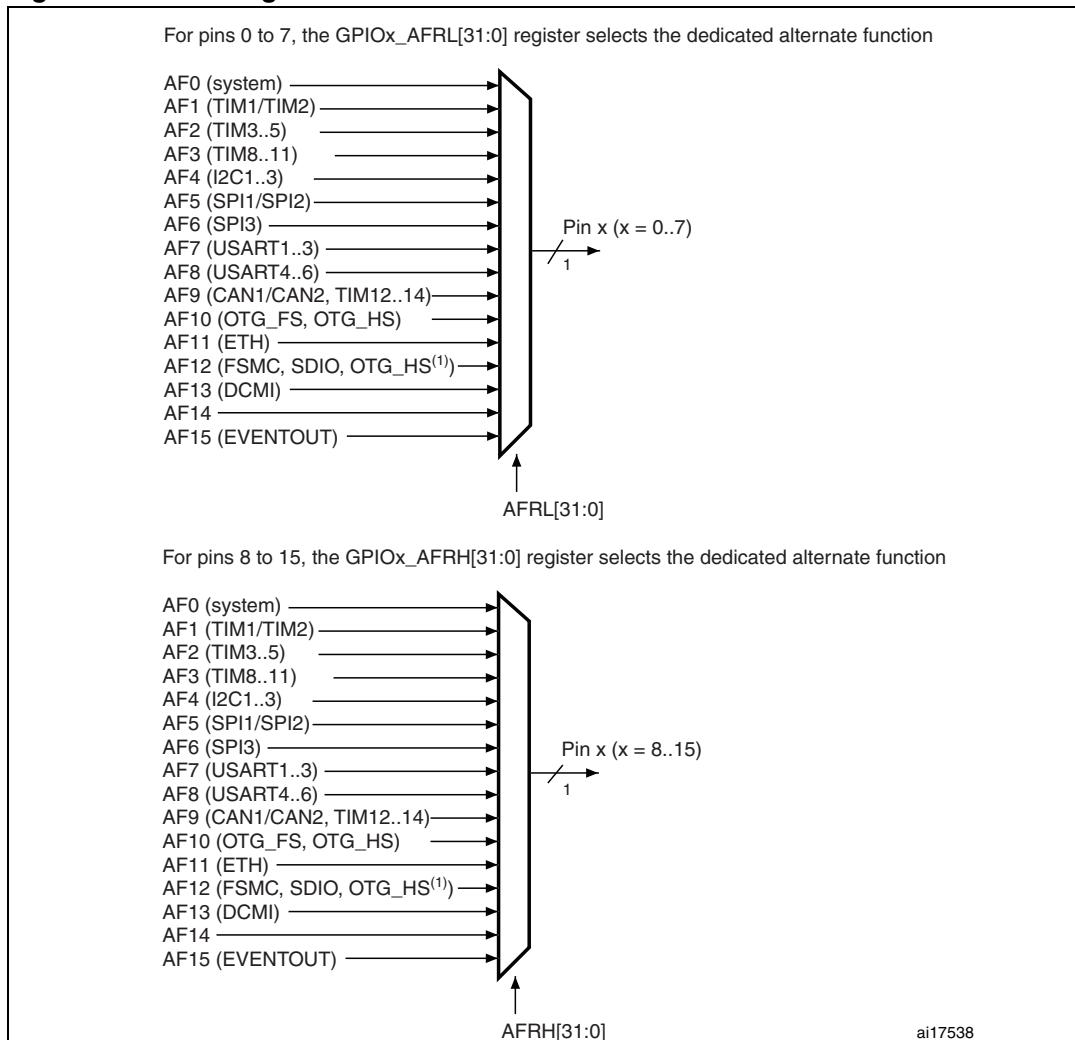
For other peripherals:

- Configure the desired I/O as an alternate function in the GPIOx_MODER register
- Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER registers, respectively
- Connect the I/O to the desired AFx in the GPIOx_AFRL or GPIOx_AFRH register

- **EVENTOUT:** you can configure the I/O pin used to output the Cortex™-M4F EVENTOUT signal by connecting it to AF15

Note: *EVENTOUT is not mapped onto the following I/O pins: PC13, PC14, PC15, PH0, PH1 and PI8.*

Please refer to the “Alternate function mapping” table in the STM32F40x and STM32F41x datasheets for the detailed mapping of the system and peripherals’ alternate function I/O pins.

Figure 14. Selecting an alternate function

1. Configured in FS.

6.3.3 I/O port control registers

Each of the GPIOs has four 32-bit memory-mapped control registers (GPIO_x_MODER, GPIO_x_OTYPER, GPIO_x_OSPEEDR, GPIO_x_PUPDR) to configure up to 16 I/Os. The GPIO_x_MODER register is used to select the I/O direction (input, output, AF, analog). The GPIO_x_OTYPER and GPIO_x_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed (the I/O speed pins are directly connected to the corresponding GPIO_x_OSPEEDR register bits whatever the I/O direction). The GPIO_x_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

6.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIO_x_IDR and GPIO_x_ODR). GPIO_x_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 (GPIO_x_IDR), a read-only register.

See [Section 6.4.5: GPIO port input data register \(GPIO_x_IDR\) \(x = A..I\)](#) and [Section 6.4.6: GPIO port output data register \(GPIO_x_ODR\) \(x = A..I\)](#) for the register descriptions.

6.3.5 I/O data bitwise handling

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

To each bit in GPIO_x_ODR, correspond two control bits in GPIO_x_BSRR: BSRR(i) and BSRR(i+SIZE). When written to 1, bit BSRR(i) **sets** the corresponding ODR(i) bit. When written to 1, bit BSRR(i+SIZE) **resets** the ODR(i) corresponding bit.

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

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

There is no need for the software to disable interrupts when programming the GPIO_x_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB1 write access.

6.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIO_x_LCKR register. The frozen registers are GPIO_x_MODER, GPIO_x_OTYPER, GPIO_x_OSPEEDR, GPIO_x_PUPDR, GPIO_x_AFRL and GPIO_x_AFRH.

To write the GPIO_x_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 GPIO_x_LCKR bit freezes the corresponding bit in the control registers (GPIO_x_MODER, GPIO_x_OTYPER, GPIO_x_OSPEEDR, GPIO_x_PUPDR, GPIO_x_AFRL and GPIO_x_AFRH).

The LOCK sequence (refer to [Section 6.4.8: GPIO port configuration lock register \(GPIO_x_LCKR\) \(x = A..I\)](#)) can only be performed using a word (32-bit long) access to the GPIO_x_LCKR register due to the fact that GPIO_x_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 6.4.8: GPIO port configuration lock register \(GPIO_x_LCKR\) \(x = A..I\)](#).

6.3.7 I/O alternate function input/output

Two registers are provided to select one out of the sixteen 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 GPIO_x_AFRL and GPIO_x_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 one I/O.

To know which functions are multiplexed on each GPIO pin, refer to the STM32F40x and STM32F41x datasheets.

Note: *The application is allowed to select one of the possible peripheral functions for each I/O at a time.*

6.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode, refer to [Section 9.2: External interrupt/event controller \(EXTI\)](#) and [Section 9.2.3: Wakeup event management](#).

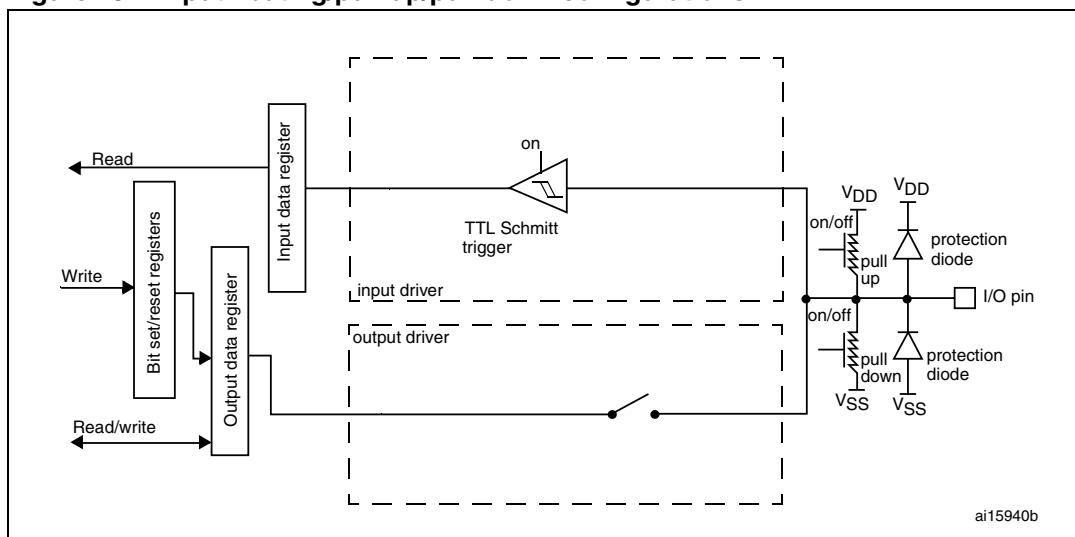
6.3.9 Input configuration

When the I/O port is programmed as Input:

- the output buffer is disabled
- the Schmitt trigger input is activated
- the pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB1 clock cycle
- A read access to the input data register provides the I/O State

[Figure 15](#) shows the input configuration of the I/O port bit.

Figure 15. Input floating/pull up/pull down configurations



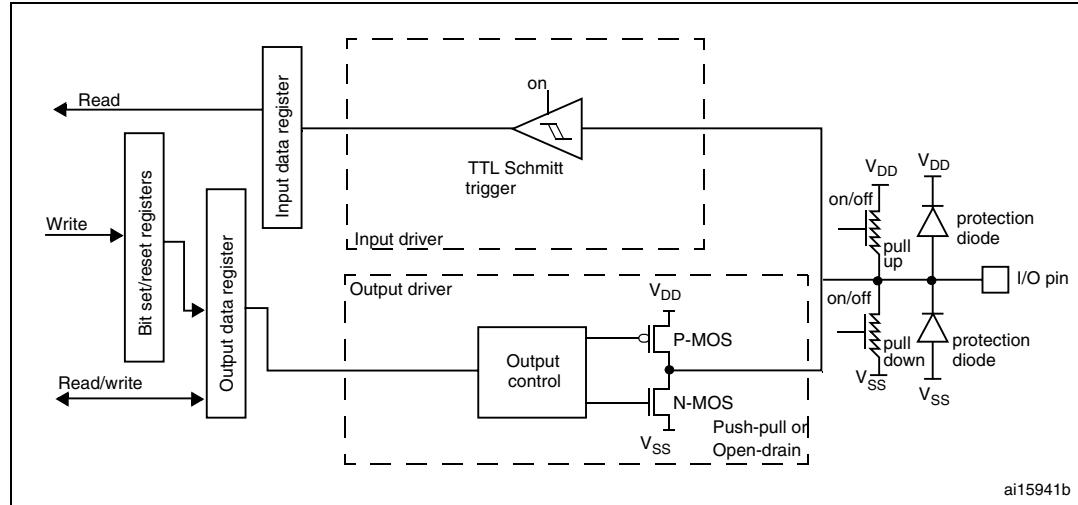
6.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
 - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
 - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The 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 AHB1 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 in Push-pull mode

Figure 16 shows the output configuration of the I/O port bit.

Figure 16. Output configuration



ai15941b

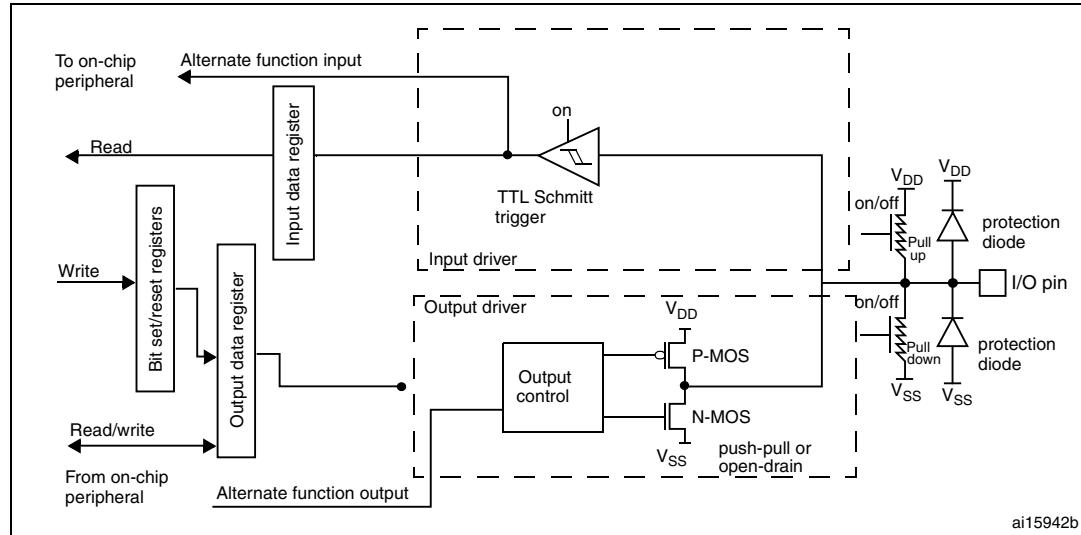
6.3.11 Alternate function configuration

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

- The output buffer is turned on in open-drain or push-pull configuration
- The output buffer is driven by the signal coming from the peripheral (alternate function out)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are 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 AHB1 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 value written in push-pull mode

Figure 17 shows the Alternate function configuration of the I/O port bit.

Figure 17. Alternate function configuration



ai15942b

6.3.12 Analog configuration

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

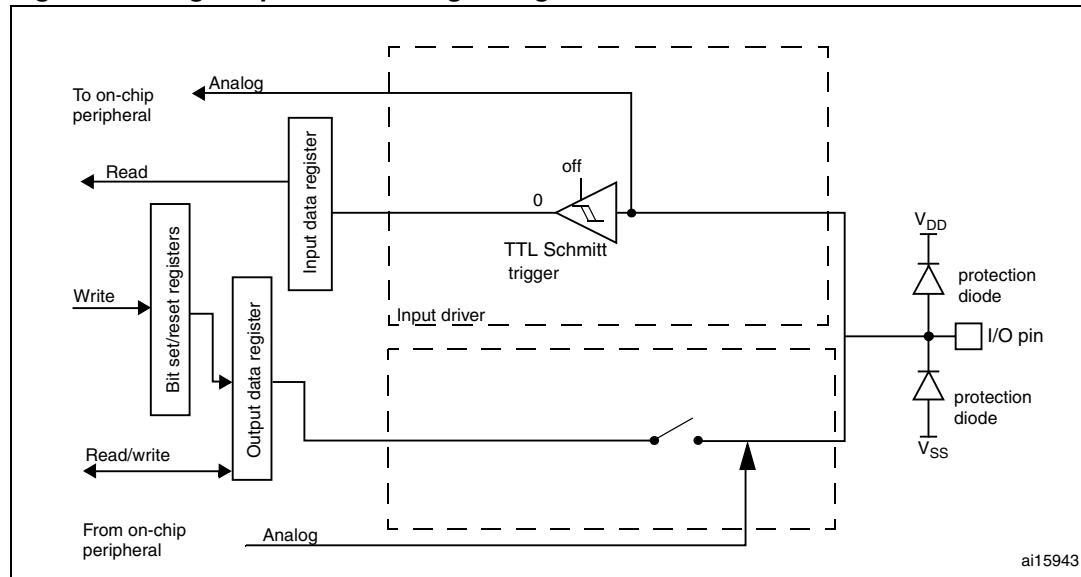
- The output buffer is disabled
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled
- Read access to the input data register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

Note:

In the analog configuration, the I/O pins cannot be 5 Volt tolerant.

Figure 18 shows the high-impedance, analog-input configuration of the I/O port bit.

Figure 18. High impedance-analog configuration



ai15943

6.3.13 Using the OSC32_IN/OSC32_OUT pins as GPIO PC14/PC15 port pins

The LSE oscillator pins OSC32_IN and OSC32_OUT can be used as general-purpose PC14 and PC15 I/Os, respectively, when the LSE oscillator is off. The PC14 and PC15 I/Os are only configured as LSE oscillator pins OSC32_IN and OSC32_OUT when the LSE oscillator is ON. This is done by setting the LSEON bit in the RCC_BDCR register. The LSE has priority over the GPIO function.

Note: *The PC14/PC15 GPIO functionality is lost when the 1.2 V domain is powered off (by the device entering the standby mode) or when the backup domain is supplied by V_{BAT} (V_{DD} no more supplied). In this case the I/Os are set in analog input mode.*

6.3.14 Using the OSC_IN/OSC_OUT pins as GPIO PH0/PH1 port pins

The HSE oscillator pins OSC_IN/OSC_OUT can be used as general-purpose PH0/PH1 I/Os, respectively, when the HSE oscillator is OFF. (after reset, the HSE oscillator is off). The PH0/PH1 I/Os are only configured as OSC_IN/OSC_OUT HSE oscillator pins when the HSE oscillator is ON. This is done by setting the HSEON bit in the RCC_CR register. The HSE has priority over the GPIO function.

6.3.15 Selection of RTC_AF1 and RTC_AF2 alternate functions

The STM32F40x and STM32F41x feature two GPIO pins RTC_AF1 and RTC_AF2 that can be used for the detection of a tamper or time stamp event, or AFO_ALARM, or AFO_CALIB RTC outputs.

The RTC_AF1 (PC13) can be used for the following purposes:

- RTC AFO_ALARM output: this output can be RTC Alarm A, RTC Alarm B or RTC Wakeup depending on the OSEL[1:0] bits in the RTC_CR register
- RTC AFO_CALIB output: this feature is enabled by setting the COE[23] in the RTC_CR register
- RTC AFI_TAMPER1: tamper event detection
- RTC AFI_TIMESTAMP: time stamp event detection

The RTC_AF2 (PI8) can be used for the following purposes:

- RTC AFI_TAMPER1: tamper event detection
- RTC AFI_TAMPER2: tamper event detection
- RTC AFI_TIMESTAMP: time stamp event detection

The selection of the corresponding pin is performed through the RTC_TAFCR register as follows:

- TAMP1INSEL is used to select which pin is used as the AFI_TAMPER1 tamper input
- TSINSEL is used to select which pin is used as the AFI_TIMESTAMP time stamp input
- ALARMOUTTYPE is used to select whether the RTC AFO_ALARM is output in push-pull or open-drain mode

The output mechanism follows the priority order listed in [Table 16](#) and [Table 17](#).

Table 16. RTC_AF1 pin ⁽¹⁾

Pin configuration and function	AFO_ALARM enabled	AFO_CALIB enabled	Tamper enabled	Time stamp enabled	TAMP1INSEL TAMPER1 pin selection	TSINSEL TIMESTAMP pin selection	ALARMOUTTYPE AFO_ALARM configuration
Alarm out output OD	1	Don't care	Don't care	Don't care	Don't care	Don't care	0
Alarm out output PP	1	Don't care	Don't care	Don't care	Don't care	Don't care	1
Calibration out output PP	0	1	Don't care	Don't care	Don't care	Don't care	Don't care
TAMPER1 input floating	0	0	1	0	0	Don't care	Don't care
TIMESTAMP and TAMPER1 input floating	0	0	1	1	0	0	Don't care
TIMESTAMP input floating	0	0	0	1	Don't care	0	Don't care
Standard GPIO	0	0	0	0	Don't care	Don't care	Don't care

1. OD: open drain; PP: push-pull.

Table 17. RTC_AF2 pin

Pin configuration and function	Tamper enabled	Time stamp enabled	TAMP1INSEL TAMPER1 pin selection	TSINSEL TIMESTAMP pin selection	ALARMOUTTYPE AFO_ALARM configuration
TAMPER1 input floating	1	0	1	Don't care	Don't care
TIMESTAMP and TAMPER1 input floating	1	1	1	1	Don't care
TIMESTAMP input floating	0	1	Don't care	1	Don't care
Standard GPIO	0	0	Don't care	Don't care	Don't care

6.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to [Table 18](#).

6.4.1 GPIO port mode register (GPIOx_MODER) (x = A..I)

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										
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										

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

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

00: Input (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode

6.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..I)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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[1:0]**: Port x configuration bits (y = 0..15)

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

0: Output push-pull (reset state)

1: Output open-drain

6.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..I)

Address offset: 0x08

Reset values:

- 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]								
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
OSPEEDR7[1:0]	OSPEEDR6[1:0]	OSPEEDR5[1:0]	OSPEEDR4[1:0]	OSPEEDR3[1:0]	OSPEEDR2[1:0]	OSPEEDR1[1:0]	OSPEEDR0[1:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

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

00: 2 MHz Low speed

01: 25 MHz Medium speed

10: 50 MHz Fast speed

11: 100 MHz High speed on 30 pF (80 MHz Output max speed on 15 pF)

6.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..I)

Address offset: 0x0C

Reset values:

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]	PUPDR14[1:0]	PUPDR13[1:0]	PUPDR12[1:0]	PUPDR11[1:0]	PUPDR10[1:0]	PUPDR9[1:0]	PUPDR8[1:0]								
rw	rw	rw	rw	rw	rw	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
PUPDR7[1:0]	PUPDR6[1:0]	PUPDR5[1:0]	PUPDR4[1:0]	PUPDR3[1:0]	PUPDR2[1:0]	PUPDR1[1:0]	PUPDR0[1:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **PUPDR_y[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

6.4.5 GPIO port input data register (GPIOx_IDR) (x = A..I)

Address offset: 0x10

Reset value: 0x0000 XXXX (where Xmeans undefined)

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

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

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

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

6.4.6 GPIO port output data register (GPIOx_ODR) (x = A..I)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0

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

Bits 15:0 **ODRy[15:0]**: Port output data (y = 0..15)

These bits can be read and written by software.

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

6.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..I)

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 and can be accessed in word, half-word or byte mode. 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 and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

6.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A..I)

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 write sequence.

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

Address offset: 0x1C

Reset value: 0x0000 0000

Access: 32-bit word only, read/write register

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

6.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A..I)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw												

Bits 31:0 **AFRLy**: Alternate function selection for port x bit y (y = 0..7)

These bits are written by software to configure alternate function I/Os

AFRLy selection:

0000: AF0

1000: AF8

0001: AF1

1001: AF9

0010: AF2

1010: AF10

0011: AF3

1011: AF11

0100: AF4

1100: AF12

0101: AF5

1101: AF13

0110: AF6

1110: AF14

0111: AF7

1111: AF15

6.4.10 GPIO alternate function high register (GPIOx_AFRH) ($x = A..I$)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
rw	rw	rw	rw												

Bits 31:0 **AFRH_y**: Alternate function selection for port x bit y (y = 8..15)

These bits are written by software to configure alternate function I/Os

AFRH_y selection:

0000: AF0	1000: AF8
0001: AF1	1001: AF9
0010: AF2	1010: AF10
0011: AF3	1011: AF11
0100: AF4	1100: AF12
0101: AF5	1101: AF13
0110: AF6	1110: AF14
0111: AF7	1111: AF15

6.4.11 GPIO register map

Table 18. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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]	
		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	
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]	
		0 0	0 0	0 0	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	GPIOx_MODER (where x = C..I)	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]	
		0 0	0 0	0 0	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..I)	Reserved												OT15	OT14	OT13	OT12
		0 1	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	

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

Offset	Register	Register Content	
0x08	GPIOx_OSPEEDER (where x = A..) except B)	OSPEEDR15[1:0]	OSPEEDR15[1:0]
		Reset value	0
0x08	GPIOB_OSPEEDER	OSPEEDR15[1:0]	OSPEEDR15[1:0]
		Reset value	0
0x0C	GPIOA_PUPDR	PUPDR15[1:0]	PUPDR15[1:0]
		Reset value	1
0x0C	GPIOB_PUPDR	PUPDR14[1:0]	PUPDR14[1:0]
		Reset value	0
0x0C	GPIOx_PUPDR (where x = C..I)	PUPDR13[1:0]	PUPDR13[1:0]
		Reset value	0
0x10	GPIOx_IDR (where x = A..I)	Reserved	
0x14	GPIOx_ODR (where x = A..I)	Reserved	
0x18	GPIOx_BSRR (where x = A..I)	Reserved	
0x1C	GPIOx_LCKR (where x = A..I)	Reserved	
0x20	GPIOx_AFRL (where x = A..I)	AFRL7[3:0]	AFRL6[3:0]
		AFRL5[3:0]	AFRL4[3:0]
0x24	GPIOx_AFRH (where x = A..I)	AFRH15[3:0]	AFRH14[3:0]
		AFRH13[3:0]	AFRH12[3:0]
	Reset value	0	0

Refer to [Table 1 on page 50](#) for the register boundary addresses. The following tables give the GPIO register map and the reset values.

7**System configuration controller (SYSCFG)**

The system configuration controller is mainly used to remap the memory accessible in the code area, select the Ethernet PHY interface and manage the external interrupt line connection to the GPIOs.

7.1 I/O compensation cell

By default the I/O compensation cell is not used. However when the I/O output buffer speed is configured in 50 MHz or 100 MHz mode, it is recommended to use the compensation cell for slew rate control on I/O $t_f(\text{IO}_{\text{out}})/t_r(\text{IO}_{\text{out}})$ commutation to reduce the I/O noise on power supply.

When the compensation cell is enabled, a READY flag is set to indicate that the compensation cell is ready and can be used. The I/O compensation cell can be used only when the supply voltage ranges from 2.4 to 3.6 V.

7.2 SYSCFG registers

7.2.1 SYSCFG memory remap register (SYSCFG_MEMRMP)

This register is used for specific configurations on memory remap:

- Two bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT pins.
- After reset these bits take the value selected by the BOOT pins. When booting from main Flash memory with BOOT pins set to 10 [(BOOT1,BOOT0) = (1,0)] this register takes the value 0x00.

When the FSMC is remapped at address 0x0000 0000, only the first two regions of Bank 1 memory controller (Bank1 NOR/PSRAM 1 and NOR/PSRAM 2) can be remapped. In remap mode, the CPU can access the external memory via ICode bus instead of System bus which boosts up the performance.

Address offset: 0x00

Reset value: 0x0000 000X (X is the memory mode selected by the BOOT pins

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

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 MEM_MODE: Memory mapping selection

Set and cleared by software. This bit controls the memory internal mapping at address 0x0000 0000. After reset these bits take on the memory mapping selected by the BOOT pins.

00: Main Flash memory mapped at 0x0000 0000

01: System Flash memory mapped at 0x0000 0000

10: FSMC Bank1 (NOR/PSRAM 1 and 2) mapped at 0x0000 0000

11: Embedded SRAM (112kB) mapped at 0x0000 0000

7.2.2 SYSCFG peripheral mode configuration register (SYSCFG_PMC)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 MII_RMII_SEL: Ethernet PHY interface selection

Set and Cleared by software. These bits control the PHY interface for the Ethernet MAC.

0: MII interface is selected

1: RMII Why interface is selected

Note: This configuration must be done while the MAC is under reset and before enabling the MAC clocks.

Bits 22:0 Reserved, must be kept at reset value.

7.2.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

Address offset: 0x08

Reset value: 0x0000

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

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

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

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

- 0000: PA[x] pin
- 0001: PB[x] pin
- 0010: PC[x] pin
- 0011: PD[x] pin
- 0100: PE[x] pin
- 0101: PF[C] pin
- 0110: PG[x] pin
- 0111: PH[x] pin
- 1000: PI[x] pin

7.2.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)

Address offset: 0x0C

Reset value: 0x0000

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

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

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

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

- 0000: PA[x] pin
- 0001: PB[x] pin
- 0010: PC[x] pin
- 0011: PD[x] pin
- 0100: PE[x] pin
- 0101: PF[x] pin
- 0110: PG[x] pin
- 0111: PH[x] pin
- 1000: PI[x] pin

7.2.5 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

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

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

- 0000: PA[x] pin
- 0001: PB[x] pin
- 0010: PC[x] pin
- 0011: PD[x] pin
- 0100: PE[x] pin
- 0101: PF[x] pin
- 0110: PG[x] pin
- 0111: PH[x] pin
- 1000: PI[x] pin

7.2.6 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)

Address offset: 0x14

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw												

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

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

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

- 0000: PA[x] pin
- 0001: PB[x] pin
- 0010: PC[x] pin
- 0011: PD[x] pin
- 0100: PE[x] pin
- 0101: PF[x] pin
- 0110: PG[x] pin
- 0111: PH[x] pin

Note: PI[15:12] are not used.

7.2.7 Compensation cell control register (SYSCFG_CMPCR)

Address offset: 0x20

Reset value: 0x0000 0000

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

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

Bit 8 **READY**: Compensation cell ready flag

0: I/O compensation cell not ready
1: O compensation cell ready

Bits 7:2 Reserved, must be kept at reset value.

Bit 0 **CMP_PD**: Compensation cell power-down

0: I/O compensation cell power-down mode
1: I/O compensation cell enabled

7.2.8 SYSCFG register maps

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

Table 19. SYSCFG register map and reset values

Refer to [Table 1 on page 50](#) for the register boundary addresses.

8 DMA controller (DMA)

8.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory and between memory and memory. Data can be quickly moved by DMA without any CPU action. This keeps CPU resources free for other operations.

The DMA controller combines a powerful dual AHB master bus architecture with independent FIFO to optimize the bandwidth of the system, based on a complex bus matrix architecture.

The two DMA controllers have 16 streams in total (8 for each controller), each dedicated to managing memory access requests from one or more peripherals. Each stream can have up to 8 channels (requests) in total. And each has an arbiter for handling the priority between DMA requests.

8.2 DMA main features

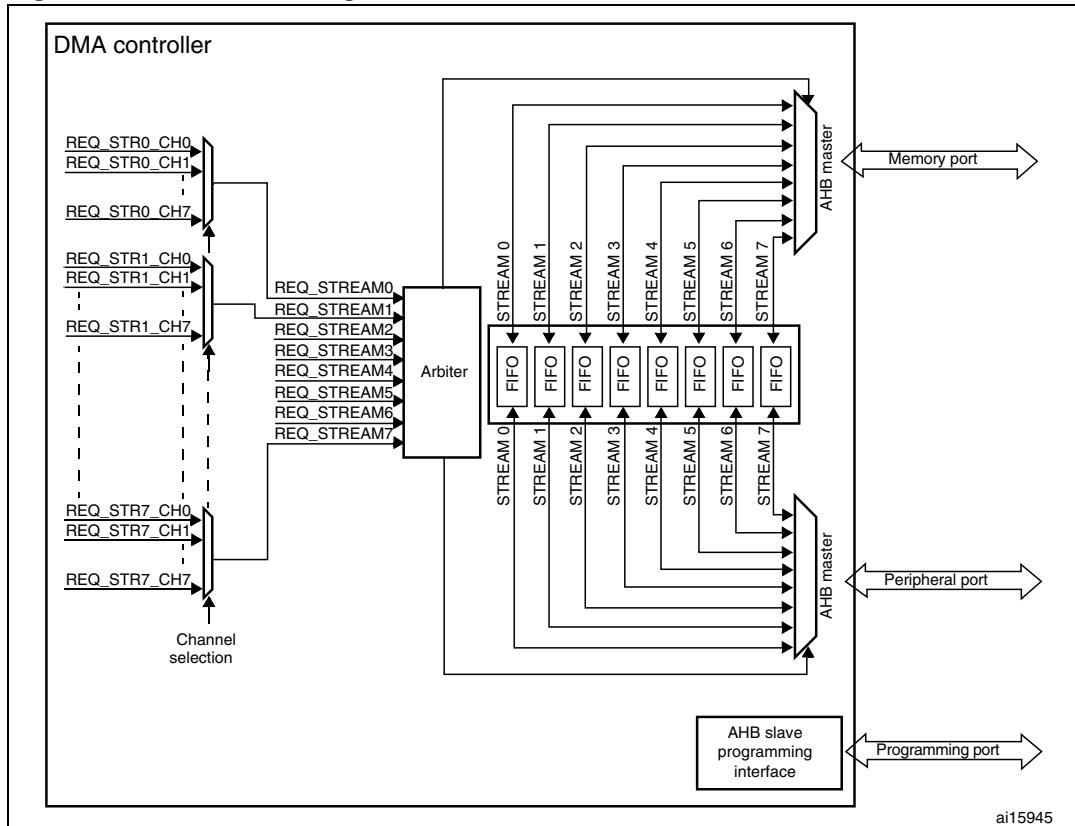
- Dual AHB master bus architecture, one dedicated to memory accesses and one dedicated to peripheral accesses
- AHB slave programming interface supporting only 32-bit accesses
- 8 streams for each DMA controller, up to 8 channels (requests) per stream
- Four separate 32 first-in, first-out memory buffers (FIFOs) per stream, that can be used in FIFO mode or direct mode:
 - FIFO mode: with threshold level software selectable between 1/4, 1/2 or 3/4 of the FIFO size
 - Direct mode: where each DMA request immediately initiates a transfer from/to the memory
- Each stream can be configured by hardware to be:
 - a regular channel that supports peripheral-to-memory, memory-to-peripheral and memory-to-memory transfers
 - a double buffer channel that also supports double buffering on the memory side
- Each of the 8 streams are connected to dedicated hardware DMA channels (requests)
- Priorities between DMA stream requests are software-programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (request 0 has priority over request 1, etc.)
- Each stream also supports software trigger for memory-to-memory transfers (only available for the DMA2 controller)
- Each stream request can be selected among up to 8 possible channel requests. This selection is software-configurable and allows several peripherals to initiate DMA requests
- The number of data items to be transferred can be managed either by the DMA controller or by the peripheral:
 - DMA flow controller: the number of data items to be transferred is software-programmable from 1 to 65535
 - Peripheral flow controller: the number of data items to be transferred is unknown and controlled by the source or the destination peripheral that signals the end of the transfer by hardware
- Independent source and destination transfer width (byte, half-word, word): when the data widths of the source and destination are not equal, the DMA automatically packs/unpacks the necessary transfers to optimize the bandwidth. This feature is only available in FIFO mode
- Incrementing or nonincrementing addressing for source and destination
- Supports incremental burst transfers of 4, 8 or 16 beats. The size of the burst is software-configurable, usually equal to half the FIFO size of the peripheral
- Each stream supports circular buffer management
- 5 event flags (DMA Half Transfer, DMA Transfer complete, DMA Transfer Error, DMA FIFO Error, Direct Mode Error) logically ORed together in a single interrupt request for each stream

8.3 DMA functional description

8.3.1 General description

Figure 19 shows the block diagram of a DMA.

Figure 19. DMA block diagram



The DMA controller performs direct memory transfer: as an AHB master, it can take the control of the AHB bus matrix to initiate AHB transactions.

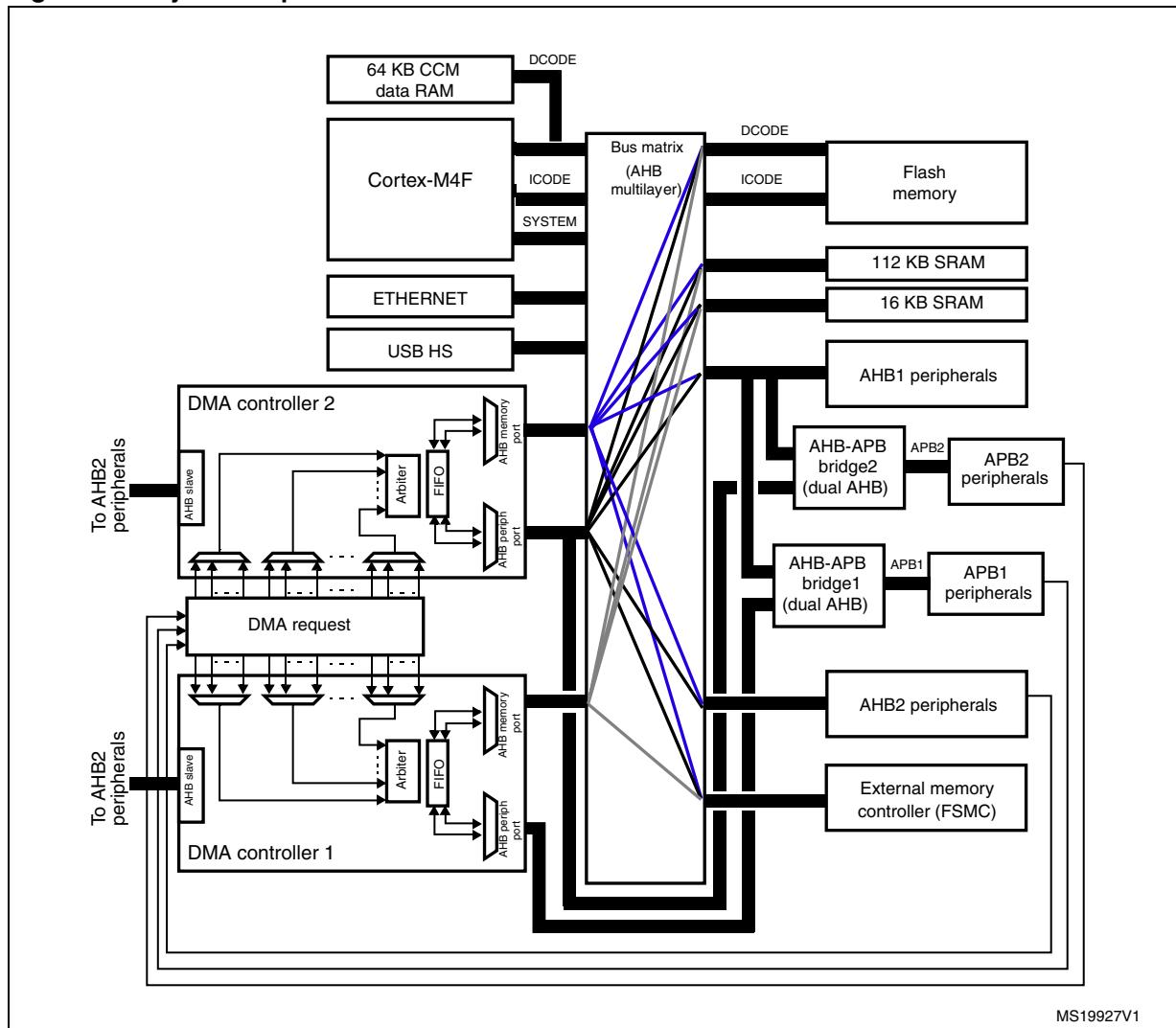
It can carry out the following transactions:

- peripheral-to-memory
- memory-to-peripheral
- memory-to-memory

The DMA controller provides two AHB master ports: the *AHB memory port*, intended to be connected to memories and the *AHB peripheral port*, intended to be connected to peripherals. However, to allow memory-to-memory transfers, the *AHB peripheral port* must also have access to the memories.

The AHB slave port is used to program the DMA controller (it supports only 32-bit accesses).

Figure 20 illustrates the implementation of the system of two DMA controllers.

Figure 20. System implementation of two DMA controllers

1. The DMA1 controller AHB peripheral port is not connected to the bus matrix like in the case of the DMA2 controller, thus only DMA2 streams are able to perform memory-to-memory transfers.

8.3.2 DMA transactions

A DMA transaction consists of a sequence of a given number of data transfers. The number of data items to be transferred and their width (8-bit, 16-bit or 32-bit) are software-programmable.

Each DMA transfer consists of three operations:

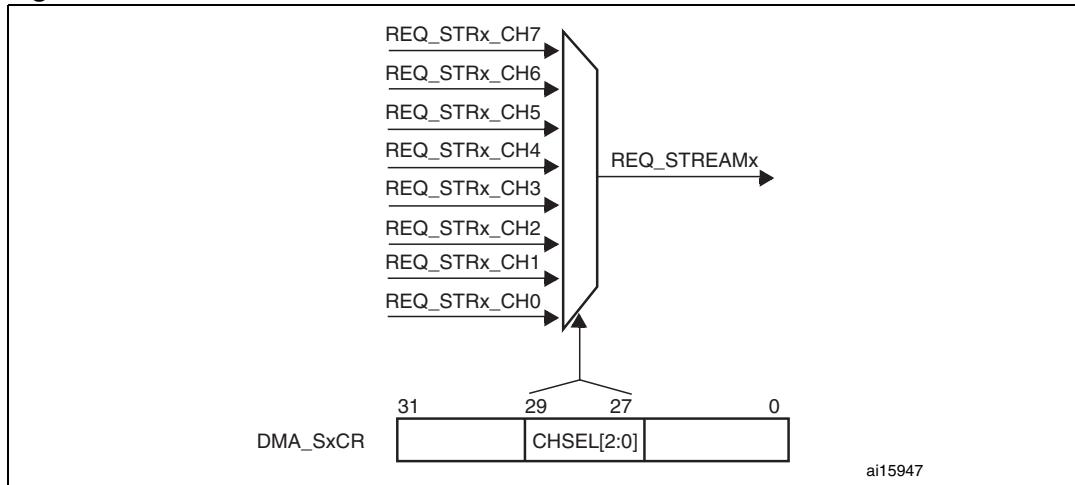
- A loading from the peripheral data register or a location in memory, addressed through the DMA_SxPAR or DMA_SxM0AR register
- A storage of the data loaded to the peripheral data register or a location in memory addressed through the DMA_SxPAR or DMA_SxM0AR register
- A post-decrement of the DMA_SxNDTR register, which contains the number of transactions that still have to be performed

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 signal is sent to the peripheral by the DMA controller. The peripheral releases its request as soon as it gets the Acknowledge signal from the DMA controller. Once the request has been deasserted by the peripheral, the DMA controller releases the Acknowledge signal. If there are more requests, the peripheral can initiate the next transaction.

8.3.3 Channel selection

Each stream is associated with a DMA request that can be selected out of 8 possible channel requests. The selection is controlled by the CHSEL[2:0] bits in the DMA_SxCR register.

Figure 21. Channel selection



The 8 requests from the peripherals (TIM, ADC, SPI, I2C, etc.) are independently connected to each channel and their connection depends on the product implementation.

Table 20 and *Table 21* give examples of DMA request mappings.

Table 20. DMA1 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	SPI3_RX		SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX		SPI3_TX
Channel 1	I2C1_RX		TIM7_UP		TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
Channel 2	TIM4_CH1		I2S2_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
Channel 3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_UP TIM2_CH4	TIM2_UP TIM2_CH4
Channel 4	UART5_RX	USART3_RX	UART4_RX	USART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
Channel 5			TIM3_CH4 TIM3_UP		TIM3_CH1 TIM3_TRIG	TIM3_CH2		TIM3_CH3
Channel 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2		TIM5_UP	
Channel 7		TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2C2_TX

Table 21. DMA2 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	ADC1		TIM8_CH1 TIM8_CH2 TIM8_CH3		ADC1		TIM1_CH1 TIM1_CH2 TIM1_CH3	
Channel 1		DCMI	ADC2	ADC2				DCMI
Channel 2	ADC3	ADC3				CRYP_OUT	CRYP_IN	HASH_IN
Channel 3	SPI1_RX		SPI1_RX	SPI1_TX		SPI1_TX		
Channel 4			USART1_RX	SDIO		USART1_RX	SDIO	USART1_TX
Channel 5		USART6_RX	USART6_RX				USART6_TX	USART6_TX
Channel 6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
Channel 7		TIM8_UP	TIM8_CH1	TIM8_CH2	TIM8_CH3			TIM8_CH4 TIM8_TRIG TIM8_COM

8.3.4 Arbiter

An arbiter manages the 8 DMA stream requests based on their priority for each of the two AHB master ports (memory and peripheral ports) and launches the peripheral/memory access sequences.

Priorities are managed in two stages:

- Software: each stream priority can be configured in the DMA_SxCR register. There are four levels:
 - Very high priority
 - High priority
 - Medium priority
 - Low priority
- Hardware: If two requests have the same software priority level, the stream with the lower number takes priority over the stream with the higher number. For example, Stream 2 takes priority over Stream 4.

8.3.5 DMA streams

Each of the 8 DMA controller streams provides a unidirectional transfer link between a source and a destination.

Each stream can be configured to perform:

- Regular type transactions: memory-to-peripherals, peripherals-to-memory or memory-to-memory transfers
- Double-buffer type transactions: double buffer transfers using two memory pointers for the memory (while the DMA is reading/writing from/to a buffer, the application can write/read to/from the other buffer).

The amount of data to be transferred (up to 65535) is programmable and related to the source width of the peripheral that requests the DMA transfer connected to the peripheral

AHB port. The register that contains the amount of data items to be transferred is decremented after each transaction.

8.3.6 Source, destination and transfer modes

Both source and destination transfers can address peripherals and memories in the entire 4 GB area, at addresses comprised between 0x0000 0000 and 0xFFFF FFFF.

The direction is configured using the DIR[1:0] bits in the DMA_SxCR register and offers three possibilities: memory-to-peripheral, peripheral-to-memory or memory-to-memory transfers. *Table 22* describes the corresponding source and destination addresses.

Table 22. Source and destination address

Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
00	Peripheral-to-memory	DMA_SxPAR	DMA_SxM0AR
01	Memory-to-peripheral	DMA_SxM0AR	DMA_SxPAR
10	Memory-to-memory	DMA_SxPAR	DMA_SxM0AR
11	reserved	-	-

When the data width (programmed in the PSIZE or MSIZE bits in the DMA_SxCR register) is a half-word or a word, respectively, the peripheral or memory address written into the DMA_SxPAR or DMA_SxM0AR/M1AR registers has to be aligned on a word or half-word address boundary, respectively.

Peripheral-to-memory mode

Figure 22 describes this mode.

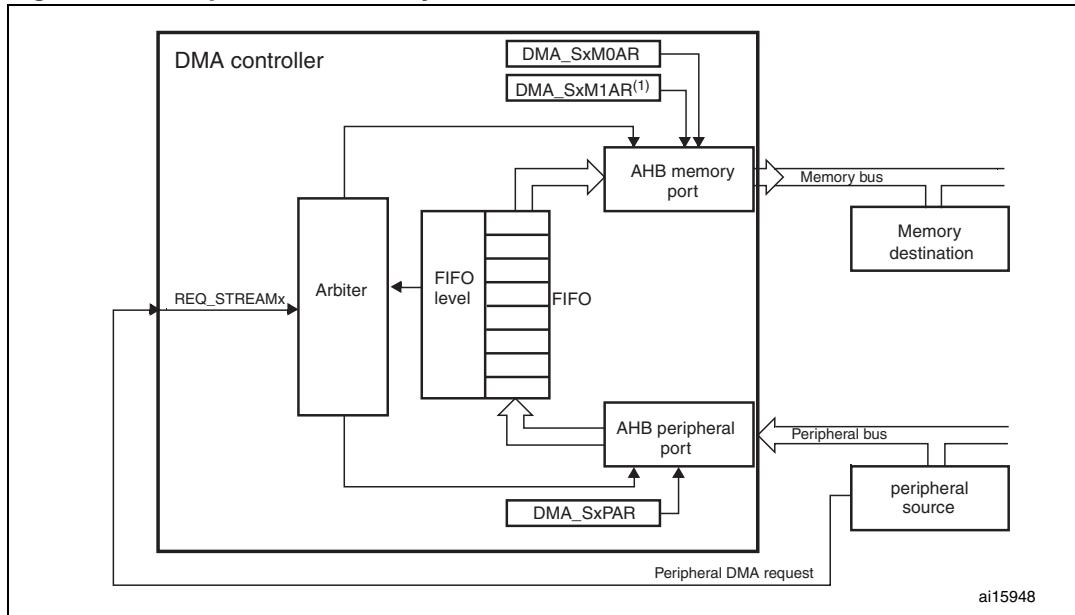
When this mode is enabled (by setting the bit EN in the DMA_SxCR register), each time a peripheral request occurs, the stream initiates a transfer from the source to fill the FIFO.

When the threshold level of the FIFO is reached, the contents of the FIFO are drained and stored into the destination.

The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In Direct mode (when the DMDIS value in the DMA_SxFcR register is '0'), the threshold level of the FIFO is not used: after each single data transfer from the peripheral to the FIFO, the corresponding data are immediately drained and stored into the destination.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Figure 22. Peripheral-to-memory mode

1. For double-buffer mode.

Memory-to-peripheral mode

Figure 23 describes this mode.

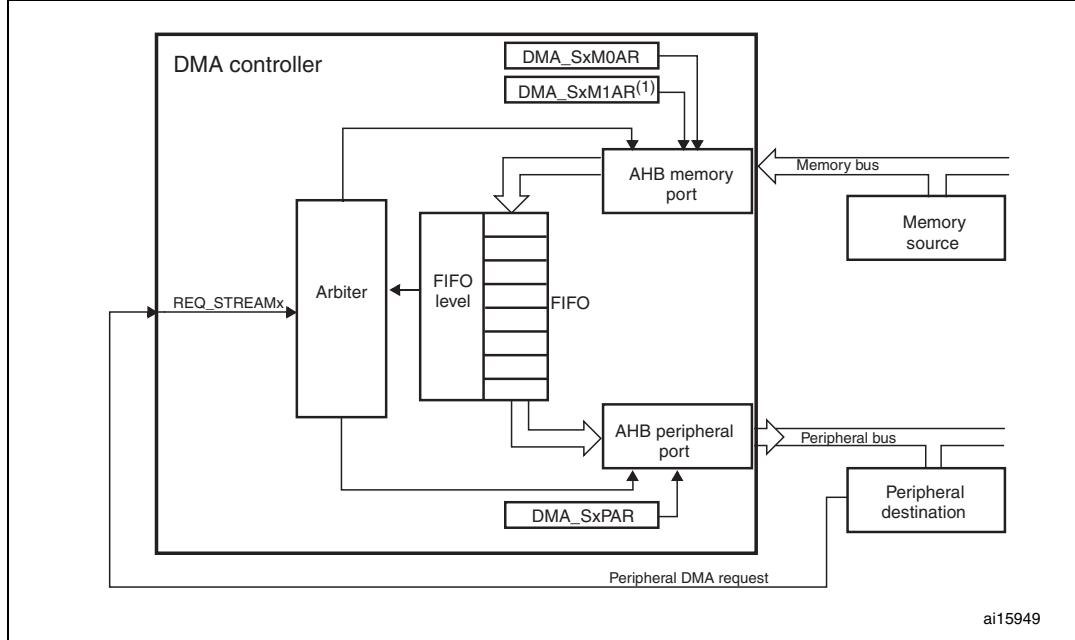
When this mode is enabled (by setting the EN bit in the DMA_SxCR register), the stream immediately initiates transfers from the source to entirely fill the FIFO.

Each time a peripheral request occurs, the contents of the FIFO are drained and stored into the destination. When the level of the FIFO is lower than or equal to the predefined threshold level, the FIFO is fully reloaded with data from the memory.

The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In Direct mode (when the DMDIS value in the DMA_SxFCR register is '0'), the threshold level of the FIFO is not used: once the stream has been enabled, only a single data transfer is initiated from the memory to the FIFO. When the corresponding peripheral transfer is complete, the FIFO is empty and the stream initiates a new single transfer from the source to the FIFO.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Figure 23. Memory-to-peripheral mode

1. For double-buffer mode.

Memory-to-memory mode

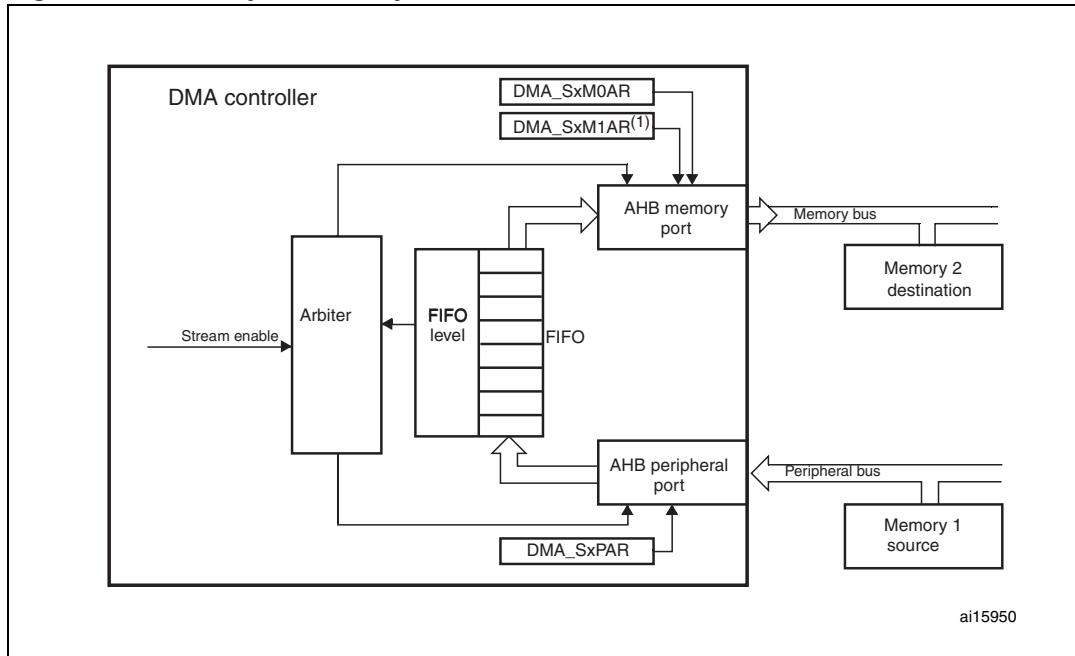
The DMA channels can also work without being triggered by a request from a peripheral. This is the memory-to-memory mode, described in [Figure 24](#).

When the stream is enabled by setting the Enable bit (EN) in the DMA_SxCR register, the stream immediately starts to fill the FIFO up to the threshold level. When the threshold level is reached, the FIFO contents are drained and stored into the destination.

The transfer stops once the DMA_SxNDTR register reaches zero or when the EN bit in the DMA_SxCR register is cleared by software.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

- Note:*
- 1 *When memory-to-memory mode is used, the Circular and Direct modes are not allowed.*
 - 2 *Only the DMA2 controller is able to perform memory-to-memory transfers.*

Figure 24. Memory-to-memory mode

1. For double-buffer mode.

8.3.7 Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented or kept constant after each transfer depending on the PINC and MINC bits in the DMA_SxCR register.

Disabling the Increment mode is useful when the peripheral source or destination data are accessed through a single register.

If the Increment mode is enabled, the address of the next transfer will be the address of the previous one incremented by 1 (for bytes), 2 (for half-words) or 4 (for words) depending on the data width programmed in the PSIZE or MSIZE bits in the DMA_SxCR register.

In order to optimize the packing operation, it is possible to fix the increment offset size for the peripheral address whatever the size of the data transferred on the AHB peripheral port. The PINCOS bit in the DMA_SxCR register is used to align the increment offset size with the data size on the peripheral AHB port, or on a 32-bit address (the address is then incremented by 4). The PINCOS bit has an impact on the AHB peripheral port only.

If PINCOS bit is set, the address of the next transfer is the address of the previous one incremented by 4 (automatically aligned on a 32-bit address) whatever the PSIZE value. The AHB memory port, however, is not impacted by this operation.

The PINC or the MINC bit needs to be set if the burst transaction is requested on the AHB peripheral port or the AHB memory port, respectively, to satisfy the AMBA protocol (burst is not allowed in the fixed address mode).

8.3.8 Circular mode

The 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_SxCR register.

When the circular mode is activated, the number of data items to be transferred is automatically reloaded with the initial value programmed during the stream configuration phase, and the DMA requests continue to be served.

Note: *In the circular mode, it is mandatory to respect the following rule in case of a burst mode configured for memory:*

$$\text{DMA_SxNDTR} = \text{Multiple of } ((\text{Mburst beat}) \times (\text{Msize})/(\text{Psize})), \text{ where:}$$

- $(\text{Mburst beat}) = 4, 8 \text{ or } 16$ (depending on the MBURST bits in the DMA_SxCR register)
- $((\text{Msize})/(\text{Psize})) = 1, 2, 4, 1/2 \text{ or } 1/4$ (Msize and Psize represent the MSIZE and PSIZE bits in the DMA_SxCR register. They are byte dependent)
- $\text{DMA_SxNDTR} = \text{Number of data items to transfer on the AHB peripheral port}$

For example: Mburst beat = 8 (INCR8), MSIZE = '00' (byte) and PSIZE = '01' (half-word), in this case: DMA_SxNDTR must be a multiple of $(8 \times 1/2 = 4)$.

If this formula is not respected, the DMA behavior and data integrity are not guaranteed.

NDTR must also be a multiple of the Peripheral burst size multiplied by the peripheral data size, otherwise this could result in a bad DMA behavior.

8.3.9 Double buffer mode

This mode is available for all the DMA1 and DMA2 streams.

The Double buffer mode is enabled by setting the DBM bit in the DMA_SxCR register.

A double-buffer stream works as a regular (single buffer) stream with the difference that it has two memory pointers. When the Double buffer mode is enabled, the Circular mode is automatically enabled (CIRC bit in DMA_SxCR is don't care) and at each end of transaction, the memory pointers are swapped.

In this mode, the DMA controller swaps from one memory target to another at each end of transaction. This allows the software to process one memory area while the second memory area is being filled/used by the DMA transfer. The double-buffer stream can work in both directions (the memory can be either the source or the destination) as described in [Table 23: Source and destination address registers in Double buffer mode \(DBM=1\)](#).

Note: *In Double buffer mode, it is possible to update the base address for the AHB memory port on-the-fly (DMA_SxM0AR or DMA_SxM1AR) when the stream is enabled, by respecting the following conditions:*

- When the CT bit is '0' in the DMA_SxCR register, the DMA_SxM1AR register can be written. Attempting to write to this register while CT = '1' sets an error flag (TEIF) and the stream is automatically disabled.
- When the CT bit is '1' in the DMA_SxCR register, the DMA_SxM0AR register can be written. Attempting to write to this register while CT = '0', sets an error flag (TEIF) and the stream is automatically disabled.

To avoid any error condition, it is advised to change the base address as soon as the TCIF flag is asserted because, at this point, the targeted memory must have changed from

memory 0 to 1 (or from 1 to 0) depending on the value of CT in the DMA_SxCR register in accordance with one of the two above conditions.

For all the other modes (except the Double buffer mode), the memory address registers are write-protected as soon as the stream is enabled.

Table 23. Source and destination address registers in Double buffer mode (DBM=1)

Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
00	Peripheral-to-memory	DMA_SxPAR	DMA_SxM0AR / DMA_SxM1AR
01	Memory-to-peripheral	DMA_SxM0AR / DMA_SxM1AR	DMA_SxPAR
10	Not allowed ⁽¹⁾		
11	Reserved	-	-

- When the Double buffer mode is enabled, the Circular mode is automatically enabled. Since the memory-to-memory mode is not compatible with the Circular mode, when the Double buffer mode is enabled, it is not allowed to configure the memory-to-memory mode.

8.3.10 Programmable data width, packing/unpacking, endianess

The number of data items to be transferred has to be programmed into DMA_SxNDTR (number of data items to transfer bit, NDT) before enabling the stream (except when the flow controller is the peripheral, PFCTRL bit in DMA_SxCR is set).

When using the internal FIFO, the data widths of the source and destination data are programmable through the PSIZE and MSIZE bits in the DMA_SxCR register (can be 8-, 16- or 32-bit).

When PSIZE and MSIZE are not equal:

- The data width of the number of data items to transfer, configured in the DMA_SxNDTR register is equal to the width of the peripheral bus (configured by the PSIZE bits in the DMA_SxCR register). For instance, in case of peripheral-to-memory, memory-to-peripheral or memory-to-memory transfers and if the PSIZE[1:0] bits are configured for half-word, the number of bytes to be transferred is equal to $2 \times NDT$.
- The DMA controller only copes with little-endian addressing for both source and destination. This is described in [Table 24: Packing/unpacking & endian behavior \(bit PINC = MINC = 1\)](#).

This packing/unpacking procedure may present a risk of data corruption when the operation is interrupted before the data are completely packed/unpacked. So, to ensure data coherence, the stream may be configured to generate burst transfers: in this case, each group of transfers belonging to a burst are indivisible (refer to [Section 8.3.11: Single and burst transfers](#)).

In Direct mode (DMDIS = 0 in the DMA_SxFIFO register), the packing/unpacking of data is not possible. In this case, it is not allowed to have different source and destination transfer data widths: both are equal and defined by the PSIZE bits in the DMA_SxCR MSIZE bits are don't care).

Table 24. Packing/unpacking & endian behavior (bit PINC = MINC = 1)

AHB memory port width	AHB peripheral port width	Number of data items to transfer (NDT)	Memory transfer number	Memory port address / byte lane	Peripheral transfer number	Peripheral port address / byte lane	
						PINCOS = 1	PINCOS = 0
8	8	4	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
8	16	2	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2	0x0 / B1IB0[15:0] 0x4 / B3IB2[15:0]	0x0 / B1IB0[15:0] 0x2 / B3IB2[15:0]
8	32	1	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1	0x0 / B3IB2IB1IB0[31:0]	0x0 / B3IB2IB1IB0[31:0]
16	8	4	1 2	0x0 / B1IB0[15:0] 0x2 / B3IB2[15:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
16	16	2	1 2	0x0 / B1IB0[15:0] 0x2 / B1IB0[15:0]	1 2	0x0 / B1IB0[15:0] 0x4 / B3IB2[15:0]	0x0 / B1IB0[15:0] 0x2 / B3IB2[15:0]
16	32	1	1 2	0x0 / B1IB0[15:0] 0x2 / B3IB2[15:0]	1	0x0 / B3IB2IB1IB0[31:0]	0x0 / B3IB2IB1IB0[31:0]
32	8	4	1	0x0 / B3IB2IB1IB0[31:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
32	16	2	1	0x0 / B3IB2IB1IB0[31:0]	1 2	0x0 / B1IB0[15:0] 0x4 / B3IB2[15:0]	0x0 / B1IB0[15:0] 0x2 / B3IB2[15:0]
32	32	1	1	0x0 / B3IB2IB1IB0[31:0]	1	0x0 / B3IB2IB1IB0[31:0]	0x0 / B3IB2IB1IB0[31:0]

Note: Peripheral port may be the source or the destination (it could also be the memory source in the case of memory-to-memory transfer).

PSIZE, MSIZE and NDT[15:0] have to be configured so as to ensure that the last transfer will not be incomplete. This can occur when the data width of the peripheral port (PSIZE bits) is lower than the data width of the memory port (MSIZE bits). This constraint is summarized in [Table 25](#).

Table 25. Restriction on NDT versus PSIZE and MSIZE

PSIZE[1:0] of DMA_SxCR	MSIZE[1:0] of DMA_SxCR	NDT[15:0] of DMA_SxNDTR
00 (8-bit)	01 (16-bit)	must be a multiple of 2
00 (8-bit)	10 (32-bit)	must be a multiple of 4
01 (16-bit)	10 (32-bit)	must be a multiple of 2

8.3.11 Single and burst transfers

The DMA controller can generate single transfers or incremental burst transfers of 4, 8 or 16 beats.

The size of the burst is configured by software independently for the two AHB ports by using the MBURST[1:0] and PBURST[1:0] bits in the DMA_SxCR register.

The burst size indicates the number of beats in the burst, not the number of bytes transferred.

To ensure data coherence, each group of transfers that form a burst are indivisible: AHB transfers are locked and the arbiter of the AHB bus matrix does not degrant the DMA master during the sequence of the burst transfer.

Depending on the single or burst configuration, each DMA request initiates a different number of transfers on the AHB peripheral port:

- When the AHB peripheral port is configured for single transfers, each DMA request generates a data transfer of a byte, half-word or word depending on the PSIZE[1:0] bits in the DMA_SxCR register
- When the AHB peripheral port is configured for burst transfers, each DMA request generates 4,8 or 16 beats of byte, half word or word transfers depending on the PBURST[1:0] and PSIZE[1:0] bits in the DMA_SxCR register.

The same as above has to be considered for the AHB memory port considering the MBURST and MSIZE bits.

In Direct mode, the stream can only generate single transfers and the MBURST[1:0] and PBURST[1:0] bits are forced by hardware.

The address pointers (DMA_SxPAR or DMA_SxM0AR registers) must be chosen so as to ensure that all transfers within a burst block are aligned on the address boundary equal to the size of the transfer.

The burst configuration has to be selected in order to respect the AHB protocol, where bursts must *not* cross the 1 KB address boundary because the minimum address space that can be allocated to a single slave is 1 KB. This means that the 1 KB address boundary should not be crossed by a burst block transfer, otherwise an AHB error would be generated, that is not reported by the DMA registers.

Note:

The Burst mode is allowed only when incrementation is enabled:

- When the PINC bit is at ‘0’, the PBURST bits should also be cleared to ‘00’
- When the MINC bit is at ‘0’, the MBURST bits should also be cleared to ‘00’.

8.3.12 FIFO

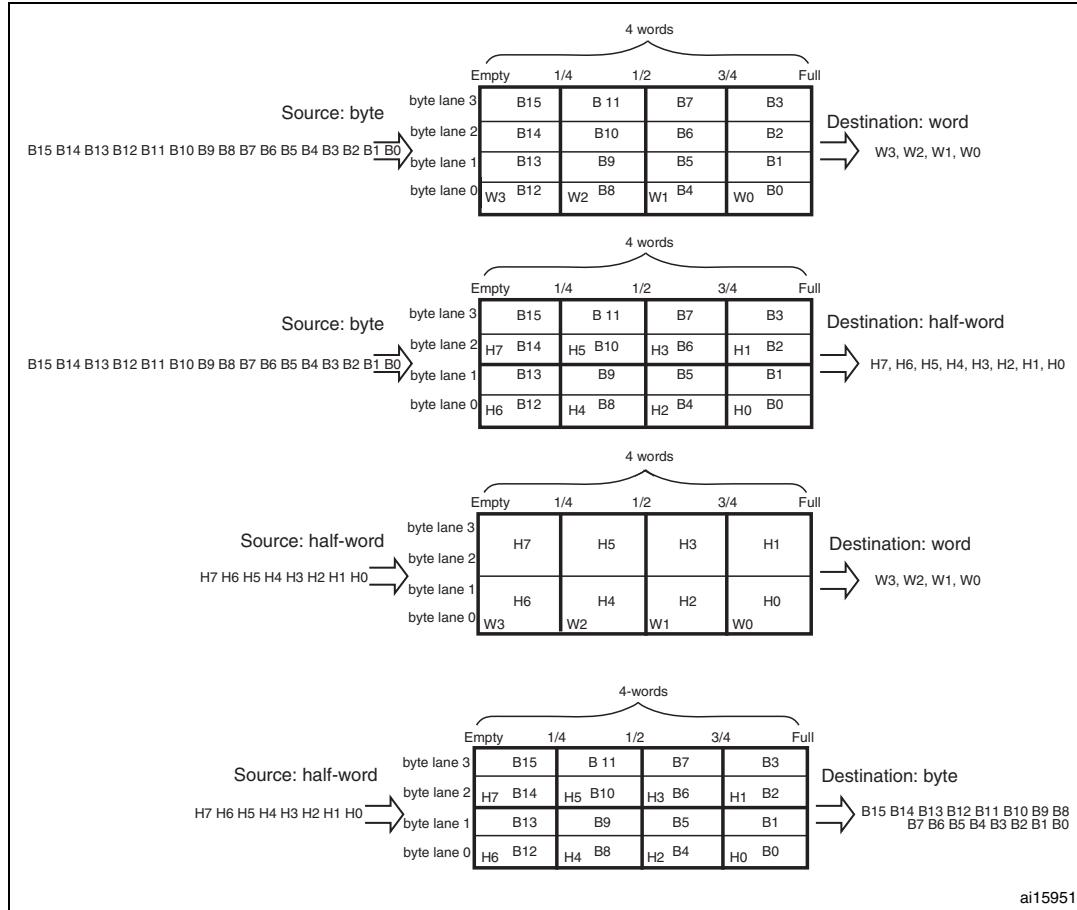
FIFO structure

The FIFO is used to temporarily store data coming from the source before transmitting them to the destination.

Each stream has an independent 4-word FIFO and the threshold level is software-configurable between 1/4, 1/2, 3/4 or full.

To enable the use of the FIFO threshold level, the Direct mode must be disabled by setting the DMDIS bit in the DMA_SxFCR register.

The structure of the FIFO differs depending on the source and destination data widths, and is described in [Figure 25: FIFO structure](#).

Figure 25. FIFO structure

ai15951

FIFO threshold and burst configuration

Caution is required when choosing the FIFO threshold (bits FTH[1:0] of the DMA_SxFCR register) and the size of the memory burst (MBURST[1:0] of the DMA_SxCR register): The content pointed by the FIFO threshold must exactly match to an integer number of memory burst transfers. If this is not in the case, a FIFO error (flag FEIFx of the DMA_HISR or DMA_LISR register) will be generated when the stream is enabled, then the stream will be automatically disabled. The allowed and forbidden configurations are described in the [Table 26: FIFO threshold configurations](#).

Table 26. FIFO threshold configurations

MSIZE	FIFO level	MBURST = INCR4	MBURST = INCR8	MBURST = INCR16
Byte	1/4	1 burst of 4 beats	forbidden	forbidden
	1/2	2 bursts of 4 beats	1 burst of 8 beats	
	3/4	3 bursts of 4 beats	forbidden	
	Full	4 bursts of 4 beats	2 bursts of 8 beats	1 burst of 16 beats

Table 26. FIFO threshold configurations (continued)

MSIZE	FIFO level	MBURST = INCR4	MBURST = INCR8	MBURST = INCR16	
Half-word	1/4	forbidden	forbidden	forbidden	
	1/2	1 burst of 4 beats			
	3/4	forbidden			
	Full	2 bursts of 4 beats	1 burst of 8 beats		
Word	1/4	forbidden	forbidden		
	1/2				
	3/4				
	Full	1 burst of 4 beats			

In all cases, the burst size multiplied by the data size must not exceed the FIFO size (data size can be: 1 (byte), 2 (half-word) or 4 (word)).

Incomplete Burst transfer at the end of a DMA transfer may happen if one of the following conditions occurs:

- For the AHB peripheral port configuration: the total number of data items (set in the DMA_SxNDTR register) is not a multiple of the burst size multiplied by the data size
- For the AHB memory port configuration: the number of remaining data items in the FIFO to be transferred to the memory is not a multiple of the burst size multiplied by the data size

In such cases, the remaining data to be transferred will be managed in single mode by the DMA, even if a burst transaction was requested during the DMA stream configuration.

Note:

When burst transfers are requested on the peripheral AHB port and the FIFO is used (DMDIS = 1 in the DMA_SxCR register), it is mandatory to respect the following rule to avoid permanent underrun or overrun conditions, depending on the DMA stream direction:

If $(PBURST \times PSIZE) = FIFO_SIZE$ (4 words), FIFO_Threshold = 3/4 is forbidden with PSIZE = 1, 2 or 4 and PBURST = 4, 8 or 16.

This rule ensures that enough FIFO space at a time will be free to serve the request from the peripheral.

FIFO flush

The FIFO can be flushed when the stream is disabled by resetting the EN bit in the DMA_SxCR register and when the stream is configured to manage peripheral-to-memory or memory-to-memory transfers: If some data are still present in the FIFO when the stream is disabled, the DMA controller continues transferring the remaining data to the destination (even though stream is effectively disabled). When this flush is completed, the transfer complete status bit (TCIFx) in the DMA_LISR or DMA_HISR register is set.

The remaining data counter DMA_SxNDTR keeps the value in this case to indicate how many data items are currently available in the destination memory.

Note that during the FIFO flush operation, if the number of remaining data items in the FIFO to be transferred to memory (in bytes) is less than the memory data width (for example 2 bytes in FIFO while MSIZE is configured to word), data will be sent with the data width set in the MSIZE bit in the DMA_SxCR register. This means that memory will be written with an

undesired value. The software may read the DMA_SxNDTR register to determine the memory area that contains the good data (start address and last address).

If the number of remaining data items in the FIFO is lower than a burst size (if the MBURST bits in DMA_SxCR register are set to configure the stream to manage burst on the AHB memory port), single transactions will be generated to complete the FIFO flush.

Direct mode

By default, the FIFO operates in Direct mode (DMDIS bit in the DMA_SxFCR is reset) and the FIFO threshold level is not used. This mode is useful when the system requires an immediate and single transfer to or from the memory after each DMA request.

To avoid saturating the FIFO, it is recommended to configure the corresponding stream with a high priority.

This mode is restricted to transfers where:

- The source and destination transfer widths are equal and both defined by the PSIZE[1:0] bits in DMA_SxCR (MSIZE[1:0] bits are don't care)
- Burst transfers are not possible (PBURST[1:0] and MBURST[1:0] bits in DMA_SxCR are don't care)

Direct mode must not be used when implementing memory-to-memory transfers.

8.3.13 DMA transfer completion

Different events can generate an end of transfer by setting the TCIFx bit in the DMA_LISR or DMA_HISR status register:

- In DMA flow controller mode:
 - The DMA_SxNDTR counter has reached zero in the memory-to-peripheral mode
 - The stream is disabled before the end of transfer (by clearing the EN bit in the DMA_SxCR register) and (when transfers are peripheral-to-memory or memory-to-memory) all the remaining data have been flushed from the FIFO into the memory
- In Peripheral flow controller mode:
 - The last external burst or single request has been generated from the peripheral and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory
 - The stream is disabled by software, and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory

Note:

The transfer completion is dependent on the remaining data in FIFO to be transferred into memory only in the case of peripheral-to-memory mode. This condition is not applicable in memory-to-peripheral mode.

If the stream is configured in noncircular mode, after the end of the transfer (that is when the number of data to be transferred reaches zero), the DMA is stopped (EN bit in DMA_SxCR register is cleared by Hardware) and no DMA request is served unless the software reprograms the stream and re-enables it (by setting the EN bit in the DMA_SxCR register).

8.3.14 DMA transfer suspension

At any time, a DMA transfer can be suspended to be restarted later on or to be definitively disabled before the end of the DMA transfer.

There are two cases:

- The stream disables the transfer with no later-on restart from the point where it was stopped. There is no particular action to do, except to clear the EN bit in the DMA_SxCR register to disable the stream. The stream may take time to be disabled (ongoing transfer is completed first). The transfer complete interrupt flag (TCIF in the DMA_LISR or DMA_HISR register) is set in order to indicate the end of transfer. The value of the EN bit in DMA_SxCR is now '0' to confirm the stream interruption. The DMA_SxNDTR register contains the number of remaining data items at the moment when the stream was stopped so that the software can determine how many data items have been transferred before the stream was interrupted.
- The stream suspends the transfer before the number of remaining data items to be transferred in the DMA_SxNDTR register reaches 0. The aim is to restart the transfer later by re-enabling the stream. In order to restart from the point where the transfer was stopped, the software has to read the DMA_SxNDTR register after disabling the stream by writing the EN bit in DMA_SxCR register (and then checking that it is at '0') to know the number of data items already collected. Then:
 - The peripheral and/or memory addresses have to be updated in order to adjust the address pointers
 - The SxNDTR register has to be updated with the remaining number of data items to be transferred (the value read when the stream was disabled)
 - The stream may then be re-enabled to restart the transfer from the point it was stopped

Note: *Note that a Transfer complete interrupt flag (TCIF in DMA_LISR or DMA_HISR) is set to indicate the end of transfer due to the stream interruption.*

8.3.15 Flow controller

The entity that controls the number of data to be transferred is known as the flow controller. This flow controller is configured independently for each stream using the PFCTRL bit in the DMA_SxCR register.

The flow controller can be:

- The DMA controller: in this case, the number of data items to be transferred is programmed by software into the DMA_SxNDTR register before the DMA stream is enabled.
- The peripheral source or destination: this is the case when the number of data items to be transferred is unknown. The peripheral indicates by hardware to the DMA controller when the last data are being transferred. This feature is only supported for peripherals which are able to signal the end of the transfer, that is:
 - SDIO

When the peripheral flow controller is used for a given stream, the value written into the DMA_SxNDTR has no effect on the DMA transfer. Actually, whatever the value written, it will

be forced by hardware to 0xFFFF as soon as the stream is enabled, to respect the following schemes:

- Anticipated stream interruption: EN bit in DMA_SxCR register is reset to 0 by the software to stop the stream before the last data hardware signal (single or burst) is sent by the peripheral. In such a case, the stream is switched off and the FIFO flush is triggered in the case of a peripheral-to-memory DMA transfer. The TCIFx flag of the corresponding stream is set in the status register to indicate the DMA completion. To know the number of data items transferred during the DMA transfer, read the DMA_SxNDTR register and apply the following formula:
– Number_of_data_transferred = 0xFFFF – DMA_SxNDTR
- Normal stream interruption due to the reception of a last data hardware signal: the stream is automatically interrupted when the peripheral requests the last transfer (single or burst) and when this transfer is complete. the TCIFx flag of the corresponding stream is set in the status register to indicate the DMA transfer completion. To know the number of data items transferred, read the DMA_SxNDTR register and apply the same formula as above.
- The DMA_SxNDTR register reaches 0: the TCIFx flag of the corresponding stream is set in the status register to indicate the forced DMA transfer completion. The stream is automatically switched off even though the last data hardware signal (single or burst) has not been yet asserted. The already transferred data will not be lost. This means that a maximum of 65535 data items can be managed by the DMA in a single transaction, even in peripheral flow control mode.

Note:

- 1 When configured in memory-to-memory mode, the DMA is always the flow controller and the PFCTRL bit is forced to 0 by hardware.
- 2 The Circular mode is forbidden in the peripheral flow controller mode.

8.3.16 Summary of the possible DMA configurations

Table 27 summarizes the different possible DMA configurations.

Table 27. Possible DMA configurations

DMA transfer mode	Source	Destination	Flow controller	Circular mode	Transfer type	Direct mode	Double buffer mode
Peripheral-to-memory	AHB peripheral port	AHB memory port	DMA	possible	single	possible	possible
					burst	forbidden	
	Peripheral		Peripheral	forbidden	single	possible	forbidden
					burst	forbidden	
Memory-to-peripheral	AHB memory port	AHB peripheral port	DMA	possible	single	possible	possible
					burst	forbidden	
	Peripheral		Peripheral	forbidden	single	possible	forbidden
					burst	forbidden	
Memory-to-memory	AHB peripheral port	AHB memory port	DMA only	forbidden	single	forbidden	forbidden
					burst		

8.3.17 Stream configuration procedure

The following sequence should be followed to configure a DMA stream x (where x is the stream number):

1. If the stream is enabled, disable it by resetting the EN bit in the DMA_SxCR register, then read this bit in order to confirm that there is no ongoing stream operation. Writing this bit to 0 is not immediately effective since it is actually written to 0 once all the current transfers have finished. When the EN bit is read as 0, this means that the stream is ready to be configured. It is therefore necessary to wait for the EN bit to be cleared before starting any stream configuration. All the stream dedicated bits set in the status register (DMA_LISR and DMA_HISR) from the previous data block DMA transfer should be cleared before the stream can be re-enabled.
2. Set the peripheral port register address in the DMA_SxPAR register. The data will be moved from/ to this address to/ from the peripheral port after the peripheral event.
3. Set the memory address in the DMA_SxMA0R register (and in the DMA_SxMA1R register in the case of a double buffer mode). The data will be written to or read from this memory after the peripheral event.
4. Configure the total number of data items to be transferred in the DMA_SxNDTR register. After each peripheral event or each beat of the burst, this value is decremented.
5. Select the DMA channel (request) using CHSEL[2:0] in the DMA_SxCR register.
6. If the peripheral is intended to be the flow controller and if it supports this feature, set the PFCTRL bit in the DMA_SxCR register.
7. Configure the stream priority using the PL[1:0] bits in the DMA_SxCR register.
8. Configure the FIFO usage (enable or disable, threshold in transmission and reception)
9. Configure the data transfer direction, peripheral and memory incremented/fixed mode, single or burst transactions, peripheral and memory data widths, Circular mode, Double buffer mode and interrupts after half and/or full transfer, and/or errors in the DMA_SxCR register.
10. Activate the stream by setting the EN bit in the DMA_SxCR register.

As soon as the stream is enabled, it can serve any DMA request from the peripheral connected to the stream.

Once half the data have been transferred on the AHB destination port, 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.

Warning: To switch off a peripheral connected to a DMA stream request, it is mandatory to, first, switch off the DMA stream to which the peripheral is connected, then to wait for EN bit = 0. Only then can the peripheral be safely disabled.

8.3.18 Error management

The DMA controller can detect the following errors:

- **Transfer error:** the transfer error interrupt flag (TEIFx) is set when:
 - A bus error occurs during a DMA read or a write access
 - A write access is requested by software on a memory address register in Double buffer mode whereas the stream is enabled and the current target memory is the one impacted by the write into the memory address register (refer to [Section 8.3.9: Double buffer mode](#))
- **FIFO error:** the FIFO error interrupt flag (FEIFx) is set if:
 - A FIFO underrun condition is detected
 - A FIFO overrun condition is detected (no detection in memory-to-memory mode because requests and transfers are internally managed by the DMA)
 - The stream is enabled while the FIFO threshold level is not compatible with the size of the memory burst (refer to [Table 26: FIFO threshold configurations](#))
- **Direct mode error:** the direct mode error interrupt flag (DMEIFx) can only be set in the peripheral-to-memory mode while operating in Direct mode and when the MINC bit in the DMA_SxCR register is cleared. This flag is set when a DMA request occurs while the previous data have not yet been fully transferred into the memory (because the memory bus was not granted). In this case, the flag indicates that 2 data items were be transferred successively to the same destination address, which could be an issue if the destination is not able to manage this situation

In Direct mode, the FIFO error flag can also be set under the following conditions:

- In the peripheral-to-memory mode, the FIFO can be saturated (overrun) if the memory bus is not granted for several peripheral requests
- In the memory-to-peripheral mode, an underrun condition may occur if the memory bus has not been granted before a peripheral request occurs

If the TEIFx or the FEIFx flag is set due to incompatibility between burst size and FIFO threshold level, the faulty stream is automatically disabled through a hardware clear of its EN bit in the corresponding stream configuration register (DMA_SxCR).

If the DMEIFx or the FEIFx flag is set due to an overrun or underrun condition, the faulty stream is not automatically disabled and it is up to the software to disable or not the stream by resetting the EN bit in the DMA_SxCR register. This is because there is no data loss when this kind of errors occur.

When the stream's error interrupt flag (TEIF, FEIF, DMEIF) in the DMA_LISR or DMA_HISR register is set, an interrupt is generated if the corresponding interrupt enable bit (TEIE, FEIE, DMIE) in the DMA_SxCR or DMA_SxFCR register is set.

Note:

When a FIFO overrun or underrun condition occurs, the data are not lost because the peripheral request is not acknowledged by the stream until the overrun or underrun condition is cleared. If this acknowledge takes too much time, the peripheral itself may detect an overrun or underrun condition of its internal buffer and data might be lost.

8.4 DMA interrupts

For each DMA stream, an interrupt can be produced on the following events:

- Half-transfer reached
- Transfer complete
- Transfer error
- Fifo error (overrun, underrun or FIFO level error)
- Direct mode error

Separate interrupt enable control bits are available for flexibility as shown in [Table 28](#).

Table 28. DMA interrupt requests

Interrupt event	Event flag	Enable control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE
FIFO overrun/underrun	FEIF	FEIE
Direct mode error	DMEIF	DMEIE

Note: Before setting an Enable control bit to '1', the corresponding event flag should be cleared, otherwise an interrupt is immediately generated.

8.5 DMA registers

Note: The DMA registers should always be accessed in word format, otherwise a bus error is generated.

8.5.1 DMA low interrupt status register (DMA_LISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
				Reserved	TCIF3	HTIF3	TEIF3	DMEIF3	Reserv ed	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Reserv ed	FEIF2
r	r	r	r	r	r	r	r	r		r	r	r	r	r	Reserv ed	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
				Reserved	TCIF1	HTIF1	TEIF1	DMEIF1	Reserv ed	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Reserv ed	FEIF0
r	r	r	r	r	r	r	r	r		r	r	r	r	r	Reserv ed	r

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **TCIFx**: Stream x transfer complete interrupt flag (x = 3..0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: No transfer complete event on stream x

1: A transfer complete event occurred on stream x

Bits 26, 20, 10, 4 **HTIFx**: Stream x half transfer interrupt flag (x=3..0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: No half transfer event on stream x

1: A half transfer event occurred on stream x

Bits 25, 19, 9, 3 **TEIFx**: Stream x transfer error interrupt flag (x=3..0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: No transfer error on stream x

1: A transfer error occurred on stream x

Bits 24, 18, 8, 2 **DMEIFx**: Stream x direct mode error interrupt flag (x=3..0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: No Direct Mode Error on stream x

1: A Direct Mode Error occurred on stream x

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **FEIFx**: Stream x FIFO error interrupt flag (x=3..0)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.

0: No FIFO Error event on stream x

1: A FIFO Error event occurred on stream x

8.5.2 DMA high interrupt status register (DMA_HISR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
				TCIF7	HTIF7	TEIF7	DMEIF7	Reserv ed	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Reserv ed	FEIF6
15	14	13	12	11	10	9	8	7	r	r	r	r	r	Reserv ed	r
				TCIF5	HTIF5	TEIF5	DMEIF5	Reserv ed	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Reserv ed	FEIF4
				r	r	r	r	Reserv ed	r	r	r	r	r	Reserv ed	r

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **TCIFx**: Stream x transfer complete interrupt flag (x=7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: No transfer complete event on stream x

1: A transfer complete event occurred on stream x

Bits 26, 20, 10, 4 **HTIFx**: Stream x half transfer interrupt flag (x=7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: No half transfer event on stream x

1: A half transfer event occurred on stream x

Bits 25, 19, 9, 3 **TEIFx**: Stream x transfer error interrupt flag (x=7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

- 0: No transfer error on stream x
- 1: A transfer error occurred on stream x

Bits 24, 18, 8, 2 **DMEIFx**: Stream x direct mode error interrupt flag (x=7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

- 0: No Direct mode error on stream x
- 1: A Direct mode error occurred on stream x

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **FEIFx**: Stream x FIFO error interrupt flag (x=7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

- 0: No FIFO error event on stream x
- 1: A FIFO error event occurred on stream x

8.5.3 DMA low interrupt flag clear register (DMA_LIFCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
				CTCIF3	CHTIF3	CTEIF3	CDMEIF3		CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2		
Reserved				rw	rw	rw	rw	Reserved	rw	rw	rw	rw	rw	Reserved	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Reserved	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Reserved	CFEIF0
				rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **CTCIFx**: Stream x clear transfer complete interrupt flag (x = 3..0)

This bit is set and cleared by software.

- 0: No effect
- 1: Clears the corresponding TCIFx flag in the DMA_LISR register

Bits 26, 20, 10, 4 **CHTIFx**: Stream x clear half transfer interrupt flag (x = 3..0)

This bit is set and cleared by software.

- 0: No effect
- 1: Clears the corresponding HTIFx flag in the DMA_LISR register

Bits 25, 19, 9, 3 **CTEIFx**: Stream x clear transfer error interrupt flag (x = 3..0)

This bit is set and cleared by software.

- 0: No effect
- 1: Clears the corresponding TEIFx flag in the DMA_LISR register

Bits 24, 18, 8, 2 **CDMEIFx**: Stream x clear direct mode error interrupt flag (x = 3..0)

This bit is set and cleared by software.

- 0: No effect
- 1: Clears the corresponding DMEIFx flag in the DMA_LISR register

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **CFEIFx**: Stream x clear FIFO error interrupt flag ($x = 3..0$)

This bit is set and cleared by software.

0: No effect

1: Clears the corresponding CFEIF x flag in the DMA_LISR register

8.5.4 DMA high interrupt flag clear register (DMA_HIFCR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Reserved	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	Reserved	CFEIF6
				rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CTCIF5	CHTIF5	CTEIF5	CDMEIF5	Reserved	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	Reserved	CFEIF4
				rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **CTCIFx**: Stream x clear transfer complete interrupt flag ($x = 7..4$)

This bit is set and cleared by software.

0: No effect

1: Clears the corresponding TCIF x flag in the DMA_HISR register

Bits 26, 20, 10, 4 **CHTIFx**: Stream x clear half transfer interrupt flag ($x = 7..4$)

This bit is set and cleared by software.

0: No effect

1: Clears the corresponding HTIF x flag in the DMA_HISR register

Bits 25, 19, 9, 3 **CTEIFx**: Stream x clear transfer error interrupt flag ($x = 7..4$)

This bit is set and cleared by software.

0: No effect

1: Clears the corresponding TEIF x flag in the DMA_HISR register

Bits 24, 18, 8, 2 **CDMEIFx**: Stream x clear direct mode error interrupt flag ($x = 7..4$)

This bit is set and cleared by software.

0: No effect

1: Clears the corresponding DMEIF x flag in the DMA_HISR register

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **CFEIFx**: Stream x clear FIFO error interrupt flag ($x = 7..4$)

This bit is set and cleared by software.

0: No effect

1: Clears the corresponding CFEIF x flag in the DMA_HISR register

8.5.5 DMA stream x configuration register (DMA_SxCR) (x = 0..7)

This register is used to configure the concerned stream.

Address offset: $0x10 + 0x18 \times \text{stream number}$

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CHSEL[3:0]			MBURST [1:0]		PBURST[1:0]		Reserved	CT	DBM or reserved	PL[1:0]	
				rw	rw	rw	rw	rw	rw	rw		rw	rw or r	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PINCO	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:25 **CHSEL[2:0]**: Channel selection

These bits are set and cleared by software.

000: channel 0 selected

001: channel 1 selected

010: channel 2 selected

011: channel 3 selected

100: channel 4 selected

101: channel 5 selected

110: channel 6 selected

111: channel 7 selected

These bits are protected and can be written only if EN is '0'

Bits 24:23 **MBURST**: Memory burst transfer configuration

These bits are set and cleared by software.

00: single transfer

01: INCR4 (incremental burst of 4 beats)

10: INCR8 (incremental burst of 8 beats)

11: INCR16 (incremental burst of 16 beats)

These bits are protected and can be written only if EN is '0'

In Direct mode, these bits are forced to 0x0 by hardware as soon as bit EN= '1'.

Bits 22:21 **PBURST[1:0]**: Peripheral burst transfer configuration

These bits are set and cleared by software.

00: single transfer

01: INCR4 (incremental burst of 4 beats)

10: INCR8 (incremental burst of 8 beats)

11: INCR16 (incremental burst of 16 beats)

These bits are protected and can be written only if EN is '0'

In Direct mode, these bits are forced to 0x0 by hardware.

Bits 20 Reserved, must be kept at reset value.

Bits 19 **CT**: Current target (only in double buffer mode)

This bits is set and cleared by hardware. It can also be written by software.

0: The current target memory is Memory 0 (addressed by the DMA_SxM0AR pointer)

1: The current target memory is Memory 1 (addressed by the DMA_SxM1AR pointer)

This bit can be written only if EN is '0' to indicate the target memory area of the first transfer.

Once the stream is enabled, this bit operates as a status flag indicating which memory area is the current target.

Bits 18 DBM: Double buffer mode

This bits is set and cleared by software.

0: No buffer switching at the end of transfer

1: Memory target switched at the end of the DMA transfer

This bit is protected and can be written only if EN is '0'.

Bits 17:16 PL[1:0]: Priority level

These bits are set and cleared by software.

00: Low

01: Medium

10: High

11: Very high

These bits are protected and can be written only if EN is '0'.

Bits 15 PINCOS: Peripheral increment offset size

This bit is set and cleared by software

0: The offset size for the peripheral address calculation is linked to the PSIZE

1: The offset size for the peripheral address calculation is fixed to 4 (32-bit alignment).

This bit has no meaning if bit PINC = '0'.

This bit is protected and can be written only if EN = '0'.

This bit is forced low by hardware when the stream is enabled (bit EN = '1') if the direct mode is selected or if PBURST are different from "00".

Bits 14:13 MSIZE[1:0]: Memory data size

These bits are set and cleared by software.

00: byte (8-bit)

01: half-word (16-bit)

10: word (32-bit)

11: reserved

These bits are protected and can be written only if EN is '0'.

In Direct mode, MSIZE is forced by hardware to the same value as PSIZE as soon as bit EN = '1'.

Bits 12:11 PSIZE[1:0]: Peripheral data size

These bits are set and cleared by software.

00: Byte (8-bit)

01: Half-word (16-bit)

10: Word (32-bit)

11: reserved

These bits are protected and can be written only if EN is '0'

Bits 10 MINC: Memory increment mode

This bit is set and cleared by software.

0: Memory address pointer is fixed

1: Memory address pointer is incremented after each data transfer (increment is done according to MSIZE)

This bit is protected and can be written only if EN is '0'.

Bits 9 PINC: Peripheral increment mode

This bit is set and cleared by software.

0: Peripheral address pointer is fixed

1: Peripheral address pointer is incremented after each data transfer (increment is done according to PSIZE)

This bit is protected and can be written only if EN is '0'.

Bits 8 **CIRC**: Circular mode

This bit is set and cleared by software and can be cleared by hardware.

0: Circular mode disabled

1: Circular mode enabled

When the peripheral is the flow controller (bit PFCTRL=1) and the stream is enabled (bit EN=1), then this bit is automatically forced by hardware to 0.

It is automatically forced by hardware to 1 if the DBM bit is set, as soon as the stream is enabled (bit EN ='1').

Bits 7:6 **DIR[1:0]**: Data transfer direction

These bits are set and cleared by software.

00: Peripheral-to-memory

01: Memory-to-peripheral

10: Memory-to-memory

11: reserved

These bits are protected and can be written only if EN is '0'.

Bits 5 **PFCTRL**: Peripheral flow controller

This bit is set and cleared by software.

0: The DMA is the flow controller

1: The peripheral is the flow controller

This bit is protected and can be written only if EN is '0'.

When the memory-to-memory mode is selected (bits DIR[1:0]=10), then this bit is automatically forced to 0 by hardware.

Bits 4 **TCIE**: Transfer complete interrupt enable

This bit is set and cleared by software.

0: TC interrupt disabled

1: TC interrupt enabled

Bits 3 **HTIE**: Half transfer interrupt enable

This bit is set and cleared by software.

0: HT interrupt disabled

1: HT interrupt enabled

Bits 2 **TEIE**: Transfer error interrupt enable

This bit is set and cleared by software.

0: TE interrupt disabled

1: TE interrupt enabled

Bits 1 **DMEIE**: Direct mode error interrupt enable

This bit is set and cleared by software.

0: DME interrupt disabled

1: DME interrupt enabled

Bits 0 **EN**: Stream enable / flag stream ready when read low

This bit is set and cleared by software.

0: Stream disabled

1: Stream enabled

This bit may be cleared by hardware:

- on a DMA end of transfer (stream ready to be configured)
- if a transfer error occurs on the AHB master buses
- when the FIFO threshold on memory AHB port is not compatible with the size of the burst

When this bit is read as 0, the software is allowed to program the Configuration and FIFO bits registers. It is forbidden to write these registers when the EN bit is read as 1.

8.5.6 DMA stream x number of data register (DMA_SxNDTR) (x = 0..7)

Address offset: $0x14 + 0x18 \times \text{stream number}$

Reset value: 0x0000 0000

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

Number of data items to be transferred (0 up to 65535). This register can be written only when the stream is disabled. When the stream is enabled, this register is read-only, indicating the remaining data items to be transmitted. This register decrements after each DMA transfer.

Once the transfer has completed, this register can either stay at zero or be reloaded automatically with the previously programmed value if the stream is configured in Circular mode.

If the value of this register is zero, no transaction can be served even if the stream is enabled.

8.5.7 DMA stream x peripheral address register (DMA_SxPAR) (x = 0..7)

Address offset: $0x18 + 0x18 \times \text{stream number}$

Reset value: 0x0000 0000

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

Bits 31:0 **PAR[31:0]**: Peripheral address

Base address of the peripheral data register from/to which the data will be read/written.

These bits are write-protected and can be written only when bit EN = '0' in the DMA_SxCR register.

8.5.8 DMA stream x memory 0 address register (DMA_SxM0AR) (x = 0..7)

Address offset: $0x1C + 0x18 \times \text{stream number}$

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[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
MOA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MOA[31:0]**: Memory 0 address

Base address of Memory area 0 from/to which the data will be read/written.

These bits are write-protected. They can be written only if:

- the stream is disabled (bit EN= '0' in the DMA_SxCR register) or
- the stream is enabled (EN='1' in DMA_SxCR register) and bit CT = '1' in the DMA_SxCR register (in Double buffer mode).

8.5.9 DMA stream x memory 1 address register (DMA_SxM1AR) (x = 0..7)

Address offset: 0x20 + 0x18 × *stream number*

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[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
M1A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **M1A[31:0]**: Memory 1 address (used in case of Double buffer mode)

Base address of Memory area 1 from/to which the data will be read/written.

This register is used only for the Double buffer mode.

These bits are write-protected. They can be written only if:

- the stream is disabled (bit EN= '0' in the DMA_SxCR register) or
- the stream is enabled (EN='1' in DMA_SxCR register) and bit CT = '0' in the DMA_SxCR register.

8.5.10 DMA stream x FIFO control register (DMA_SxFCR) (x = 0..7)

Address offset: $0x24 + 0x24 \times \text{stream number}$

Reset value: 0x0000 0021

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

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

Bits 7 **FEIE**: FIFO error interrupt enable

This bit is set and cleared by software.

- 0: FE interrupt disabled
- 1: FE interrupt enabled

Bits 6 Reserved, must be kept at reset value.

Bits 5:3 **FS[2:0]**: FIFO status

These bits are read-only.

- 000: $0 < \text{fifo_level} < 1/4$
- 001: $1/4 \leq \text{fifo_level} < 1/2$
- 010: $1/2 \leq \text{fifo_level} < 3/4$
- 011: $3/4 \leq \text{fifo_level} < \text{full}$
- 100: FIFO is empty
- 101: FIFO is full
- others: no meaning

These bits are not relevant in the Direct mode (DMDIS bit is zero).

Bits 2 **DMDIS**: Direct mode disable

This bit is set and cleared by software. It can be set by hardware.

- 0: Direct mode enabled
- 1: Direct mode disabled

This bit is protected and can be written only if EN is '0'.

This bit is set by hardware if the memory-to-memory mode is selected (DIR bit in DMA_SxCR are "10") and the EN bit in the DMA_SxCR register is '1' because the Direct mode is not allowed in the memory-to-memory configuration.

Bits 1:0 **FTH[1:0]**: FIFO threshold selection

These bits are set and cleared by software.

- 00: 1/4 full FIFO
- 01: 1/2 full FIFO
- 10: 3/4 full FIFO
- 11: full FIFO

These bits are not used in the Direct Mode when the DMIS value is zero.

These bits are protected and can be written only if EN is '1'.

8.5.11 DMA register map

Table 29 summarizes the DMA registers.

Table 29. DMA register map and reset values

Offset	Register	31	30	29	28	27	26	25	24
0x0000	DMA_LISR	Reserved	Reserved	Reserved	Reserved	TCIF3	HTIF3	TEIF3	DMEIF3
	Reset value								
0x0004	DMA_HISR	Reserved	Reserved	Reserved	Reserved	TCIF7	HTIF7	TEIF7	DMEIF7
	Reset value								
0x0008	DMA_LIFCR	Reserved	Reserved	Reserved	Reserved	CTCIF3	CHTIF3	CTEIF3	CDMEIF3
	Reset value								
0x000C	DMA_HIFCR	Reserved	Reserved	Reserved	Reserved	CTCIF7	CHTIF7	CTEIF7	CDMEIF7
	Reset value								
0x0010	DMA_S0CR	Reserved	Reserved	PBURST[1:0]	CHSEL[2:0]	MBURST[1:0]	PINCOS	MSIZE[1:0]	Reserved
	Reset value								
0x0014	DMA_S0NDTR	Reserved	Reserved	PSIZE[1:0]	MINC	PINC	CIRC	DIR[1:0]	Reserved
	Reset value								
0x0018	DMA_S0PAR	PA[31:0]	PA[31:0]	PSIZE[1:0]	CTCIF5	CHTIF5	CTEIF5	CDMEIF5	DMEIF5
	Reset value								
0x001C	DMA_S0M0AR	MOA[31:0]	MOA[31:0]	PSIZE[1:0]	CTCIF1	CHTIF1	CTEIF1	CDMEIF1	DMEIF1
	Reset value								
0x0020	DMA_S0M1AR	M1A[31:0]	M1A[31:0]	PSIZE[1:0]	PFCTRL	TCIE	HTIE	TEIE	CDMEIF4
	Reset value								
0x0024	DMA_S0FCR	Reserved	Reserved	FS[2:0]	TCIF4	HTIF4	TEIF4	CDMEIF0	DMEIF0
	Reset value								
0x0028	DMA_S1CR	Reserved	Reserved	DIR[1:0]	PFCTRL	TCIE	HTIE	TEIE	CDMEIF4
	Reset value								
0x002C	DMA_S1NDTR	Reserved	Reserved	DIR[1:0]	TCIF4	HTIF4	TEIF4	CDMEIF0	DMEIF0
	Reset value								
0x0030	DMA_S1PAR	PA[31:0]	PA[31:0]	DIR[1:0]	TCIF4	HTIF4	TEIF4	CDMEIF0	DMEIF0
	Reset value								

Table 29. 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									
0x0034	DMA_S1M0AR																																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
0x0038	DMA_S1M1AR																																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
0x003C	DMA_S1FCR																														FTH[1:0]											
	Reset value																														FEIE	FS[2:0]	DMDIS	0	0							
0x0040	DMA_S2CR	Reserved	CHSEL[2:0]		MBURST[1:0]		PBURST[1:0]		ACK		CT		DBM		PL[1:0]		PINCOS		MSIZE[1:0]		PSIZE[1:0]		MINC		PINC		CIRC		DIR[1:0]		PFCTRL		TCIE		HTIE		TEIE		DMEIE		EN	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x0044	DMA_S2NDTR	Reserved																					NDT[15:]							0	0	0	0	0								
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x0048	DMA_S2PAR																						PA[31:0]							0	0	0	0	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								
0x004C	DMA_S2M0AR																						MOA[31:0]							0	0	0	0	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								
0x0050	DMA_S2M1AR																						M1A[31:0]							0	0	0	0	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								
0x0054	DMA_S2FCR	Reserved																					Reserved							0	0	0	0	0								
	Reset value																						0	FEIE	FS[2:0]	DMDIS	0	0	0	0	0	0	0	0	0	0	0					
0x0058	DMA_S3CR	Reserved	CHSEL[2:0]		MBURST[1:0]		PBURST[1:0]		ACK		CT		DBM		PL[1:0]		PINCOS		MSIZE[1:0]		PSIZE[1:0]		MINC		PINC		CIRC		DIR[1:0]		PFCTRL		TCIE		HTIE		TEIE		DMEIE		EN	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x005C	DMA_S3NDTR	Reserved																					NDT[15:]							0	0	0	0	0								
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x0060	DMA_S3PAR																						PA[31:0]							0	0	0	0	0								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x0064	DMA_S3M0AR																						MOA[31:0]							0	0	0	0	0								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x0068	DMA_S3M1AR																						M1A[31:0]							0	0	0	0	0								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x006C	DMA_S3FCR	Reserved																					Reserved							0	FEIE	FS[2:0]	DMDIS	FTH[1:0]								
	Reset value																						1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 29. DMA register map and reset values (continued)

Table 29. DMA register map and reset values (continued)

Refer to [Table 1 on page 50](#) for the register boundary addresses.

9 Interrupts and events

This Section applies to the whole STM32F40x and STM32F41x family, unless otherwise specified.

9.1 Nested vectored interrupt controller (NVIC)

9.1.1 NVIC features

The nested vector interrupt controller NVIC includes the following features:

- 87 maskable interrupt channels (not including the 16 interrupt lines of Cortex™-M4F)
- 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 see *Chapter 5: Exceptions & Chapter 8: Nested Vectored Interrupt Controller* in the ARM Cortex™-M4F Technical Reference Manual.

9.1.2 SysTick calibration value register

The SysTick calibration value is fixed to 15000, which gives a reference time base of 1 ms with the SysTick clock set to 15 MHz (max HCLK/8).

9.1.3 Interrupt and exception vectors

Table 30 is the vector table for the STM32F40x and STM32F41x devices.

Table 30. 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

Table 30. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000_001C - 0x0000_002B
	3	settable	SVCall	System service call via SWI instruction	0x0000_002C
	4	settable	Debug Monitor	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	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 detection interrupt	0x0000_0044
2	9	settable	TAMP_STAMP	Tamper andTimeStamp interrupts through the EXTI line	0x0000_0048
3	10	settable	RTC_WKUP	RTC Wakeup interrupt through the EXTI line	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Stream0	DMA1 Stream0 global interrupt	0x0000_006C
12	19	settable	DMA1_Stream1	DMA1 Stream1 global interrupt	0x0000_0070
13	20	settable	DMA1_Stream2	DMA1 Stream2 global interrupt	0x0000_0074
14	21	settable	DMA1_Stream3	DMA1 Stream3 global interrupt	0x0000_0078
15	22	settable	DMA1_Stream4	DMA1 Stream4 global interrupt	0x0000_007C
16	23	settable	DMA1_Stream5	DMA1 Stream5 global interrupt	0x0000_0080
17	24	settable	DMA1_Stream6	DMA1 Stream6 global interrupt	0x0000_0084
18	25	settable	ADC	ADC1, ADC2 and ADC3 global interrupts	0x0000_0088
19	26	settable	CAN1_TX	CAN1 TX interrupts	0x0000_008C
20	27	settable	CAN1_RX0	CAN1 RX0 interrupts	0x0000_0090
21	28	settable	CAN1_RX1	CAN1 RX1 interrupt	0x0000_0094
22	29	settable	CAN1_SCE	CAN1 SCE interrupt	0x0000_0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C

Table 30. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
24	31	settable	TIM1_BRK_TIM9	TIM1 Break interrupt and TIM9 global interrupt	0x0000_00A0
25	32	settable	TIM1_UP_TIM10	TIM1 Update interrupt and TIM10 global interrupt	0x0000_00A4
26	33	settable	TIM1_TRG_COM_TIM11	TIM1 Trigger and Commutation interrupts and TIM11 global interrupt	0x0000_00A8
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I ² C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I ² C1 error interrupt	0x0000_00C0
33	40	settable	I2C2_EV	I ² C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I ² C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
41	48	settable	RTC_Alarm	RTC Alarms (A and B) through EXTI line interrupt	0x0000_00E4
42	49	settable	OTG_FS_WKUP	USB On-The-Go FS Wakeup through EXTI line interrupt	0x0000_00E8
43	50	settable	TIM8_BRK_TIM12	TIM8 Break interrupt and TIM12 global interrupt	0x0000_00EC
44	51	settable	TIM8_UP_TIM13	TIM8 Update interrupt and TIM13 global interrupt	0x0000_00F0
45	52	settable	TIM8_TRG_COM_TIM14	TIM8 Trigger and Commutation interrupts and TIM14 global interrupt	0x0000_00F4
46	53	settable	TIM8_CC	TIM8 Capture Compare interrupt	0x0000_00F8
47	54	settable	DMA1_Stream7	DMA1 Stream7 global interrupt	0x0000_00FC
48	55	settable	FSMC	FSMC global interrupt	0x0000_0100
49	56	settable	SDIO	SDIO global interrupt	0x0000_0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000_0108

Table 30. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
51	58	settable	SPI3	SPI3 global interrupt	0x0000_010C
52	59	settable	UART4	UART4 global interrupt	0x0000_0110
53	60	settable	UART5	UART5 global interrupt	0x0000_0114
54	61	settable	TIM6_DAC	TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts	0x0000_0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000_011C
56	63	settable	DMA2_Stream0	DMA2 Stream0 global interrupt	0x0000_0120
57	64	settable	DMA2_Stream1	DMA2 Stream1 global interrupt	0x0000_0124
58	65	settable	DMA2_Stream2	DMA2 Stream2 global interrupt	0x0000_0128
59	66	settable	DMA2_Stream3	DMA2 Stream3 global interrupt	0x0000_012C
60	67	settable	DMA2_Stream4	DMA2 Stream4 global interrupt	0x0000_0130
61	68	settable	ETH	Ethernet global interrupt	0x0000_0134
62	69	settable	ETH_WKUP	Ethernet Wakeup through EXTI line interrupt	0x0000_0138
63	70	settable	CAN2_TX	CAN2 TX interrupts	0x0000_013C
64	71	settable	CAN2_RX0	CAN2 RX0 interrupts	0x0000_0140
65	72	settable	CAN2_RX1	CAN2 RX1 interrupt	0x0000_0144
66	73	settable	CAN2_SCE	CAN2 SCE interrupt	0x0000_0148
67	74	settable	OTG_FS	USB On The Go FS global interrupt	0x0000_014C
68	75	settable	DMA2_Stream5	DMA2 Stream5 global interrupt	0x0000_0150
69	76	settable	DMA2_Stream6	DMA2 Stream6 global interrupt	0x0000_0154
70	77	settable	DMA2_Stream7	DMA2 Stream7 global interrupt	0x0000_0158
71	78	settable	USART6	USART6 global interrupt	0x0000_015C
72	79	settable	I2C3_EV	I ² C3 event interrupt	0x0000_0160
73	80	settable	I2C3_ER	I ² C3 error interrupt	0x0000_0164
74	81	settable	OTG_HS_EP1_OUT	USB On The Go HS End Point 1 Out global interrupt	0x0000_0168
75	82	settable	OTG_HS_EP1_IN	USB On The Go HS End Point 1 In global interrupt	0x0000_016C
76	83	settable	OTG_HS_WKUP	USB On The Go HS Wakeup through EXTI interrupt	0x0000_0170
77	84	settable	OTG_HS	USB On The Go HS global interrupt	0x0000_0174
78	85	settable	DCMI	DCMI global interrupt	0x0000_0178
79	86	settable	CRYP	CRYP crypto global interrupt	0x0000_017C

Table 30. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
80	87	settable	HASH_RNG	Hash and Rng global interrupt	0x0000_0180
81	88	settable	FPU	FPU global interrupt	0x0000_0184

9.2 External interrupt/event controller (EXTI)

The external interrupt/event controller consists of up to 23 edge detectors for generating event/interrupt requests. Each input line can be independently configured to select the type (pulse or pending) and the corresponding trigger event (rising or falling or both). Each line can also be masked independently. A pending register maintains the status line of the interrupt requests

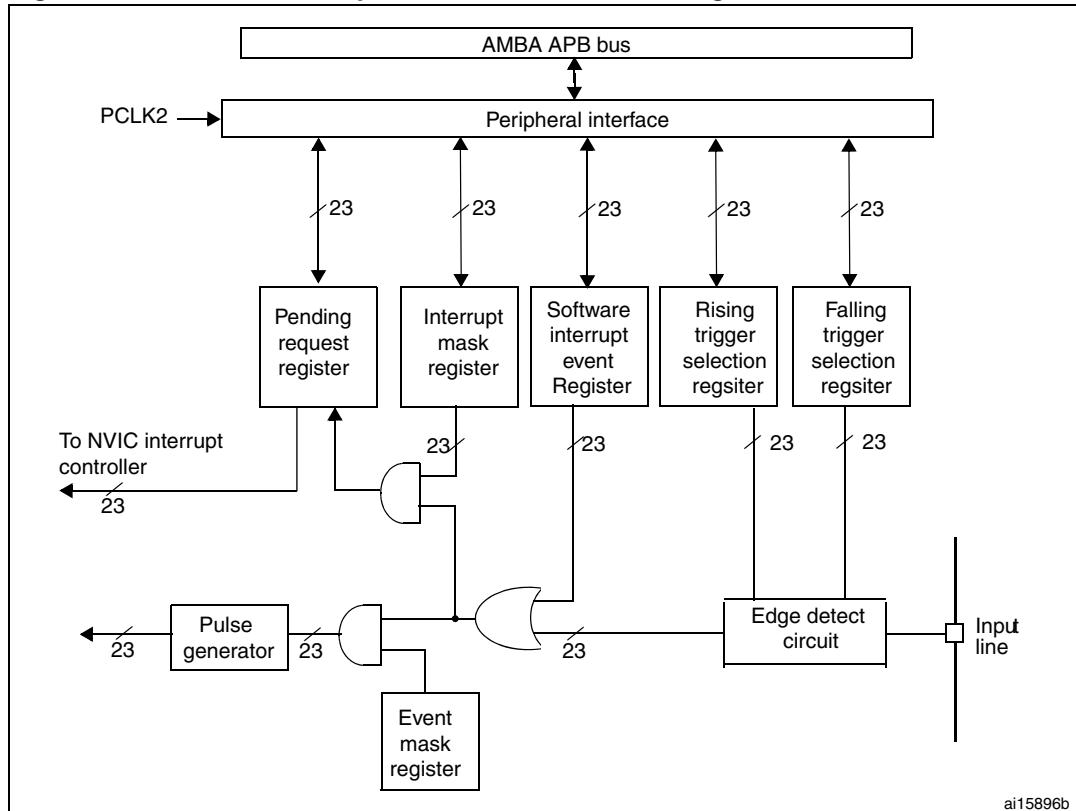
9.2.1 EXTI main features

The main features of the EXTI controller are the following:

- independent trigger and mask on each interrupt/event line
- dedicated status bit for each interrupt line
- generation of up to 23 software event/interrupt requests
- detection of external signals with a pulse width lower than the APB2 clock period. Refer to the electrical characteristics section of the STM32F40x and STM32F41x datasheets for details on this parameter.

9.2.2 EXTI block diagram

Figure 26 shows the block diagram.

Figure 26. External interrupt/event controller block diagram

9.2.3 Wakeup event management

The STM32F40x and STM32F41x 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™-M4F System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

To use an external line as a wakeup event, refer to [Section 9.2.4: Functional description](#).

9.2.4 Functional description

To generate the interrupt, the interrupt line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a ‘1’ to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is reset by writing a ‘1’ in the pending register.

To generate the event, the event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the

event request by writing a ‘1’ to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event pulse is generated. The pending bit corresponding to the event line is not set.

An interrupt/event request can also be generated by software by writing a ‘1’ in the software interrupt/event register.

Hardware interrupt selection

To configure the 23 lines as interrupt sources, use the following procedure:

- Configure the mask bits of the 23 interrupt lines (EXTI_IMR)
- Configure the Trigger selection bits of the interrupt lines (EXTI_RTSR and EXTI_FTSR)
- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the external interrupt controller (EXTI) so that an interrupt coming from one of the 23 lines can be correctly acknowledged.

Hardware event selection

To configure the 23 lines as event sources, use the following procedure:

- Configure the mask bits of the 23 event lines (EXTI_EMR)
- Configure the Trigger selection bits of the event lines (EXTI_RTSR and EXTI_FTSR)

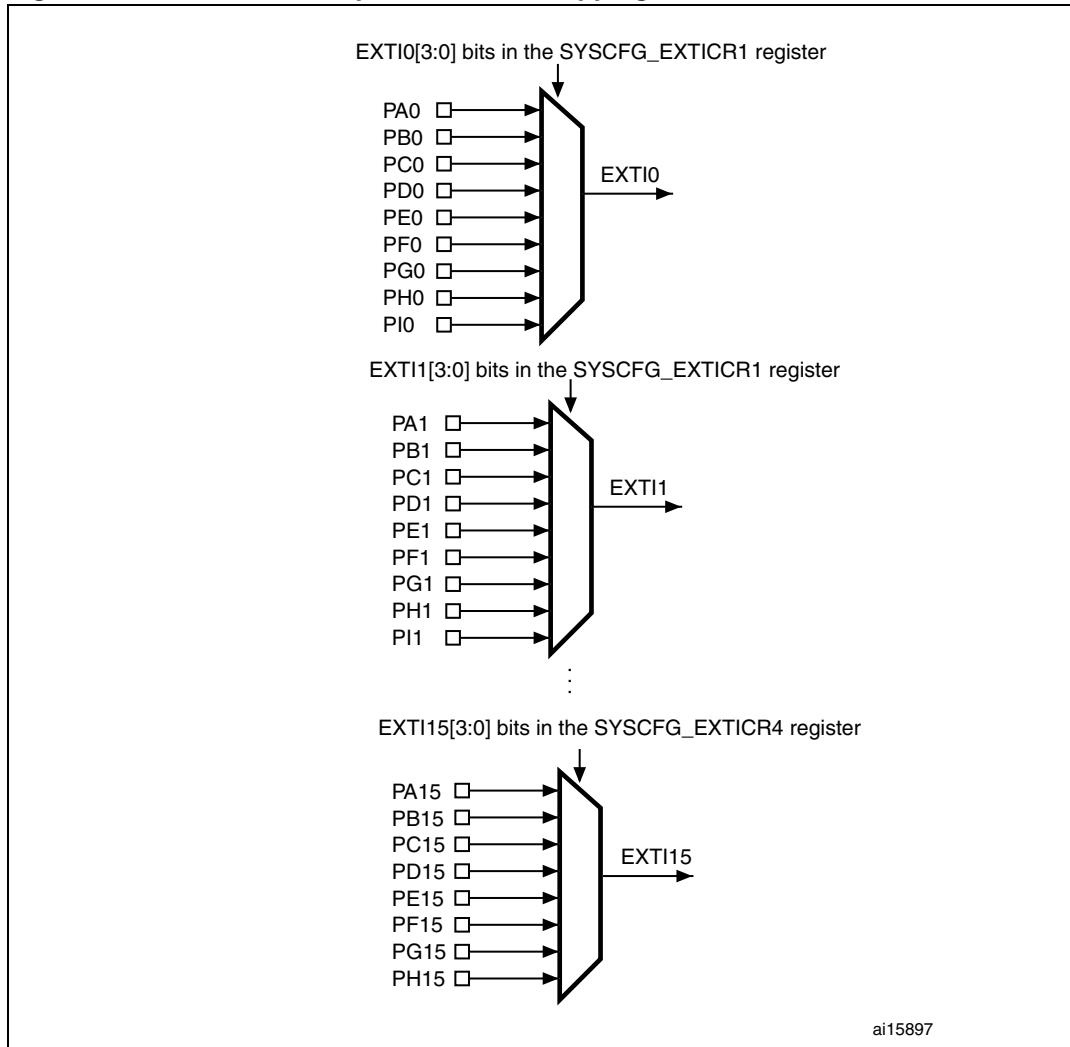
Software interrupt/event selection

The 23 lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt.

- Configure the mask bits of the 23 interrupt/event lines (EXTI_IMR, EXTI_EMR)
- Set the required bit in the software interrupt register (EXTI_SWIER)

9.2.5 External interrupt/event line mapping

The 140 GPIOs are connected to the 16 external interrupt/event lines in the following manner:

Figure 27. External interrupt/event GPIO mapping

The seven other EXTI lines are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event
- EXTI line 18 is connected to the USB OTG FS Wakeup event
- EXTI line 19 is connected to the Ethernet Wakeup event
- EXTI line 20 is connected to the USB OTG HS (configured in FS) Wakeup event
- EXTI line 21 is connected to the RTC Tamper and TimeStamp events
- EXTI line 22 is connected to the RTC Wakeup event

9.3 EXTI registers

Refer to [Section 1.1 on page 46](#) for a list of abbreviations used in register descriptions.

9.3.1 Interrupt mask register (EXTI_IMR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16					
Reserved														MR22	MR21	MR20	MR19	MR18	MR17	MR16
									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					
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					

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

Bits 22:0 **MRx**: Interrupt mask on line x

- 0: Interrupt request from line x is masked
- 1: Interrupt request from line x is not masked

9.3.2 Event mask register (EXTI_EMR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16					
Reserved														MR22	MR21	MR20	MR19	MR18	MR17	MR16
									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					
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					

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

Bits 22:0 **MRx**: Event mask on line x

- 0: Event request from line x is masked
- 1: Event request from line x is not masked

9.3.3 Rising trigger selection register (EXTI_RTSR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														TR22	TR21
									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 31:23 Reserved, must be kept at reset value.

Bits 22:0 **TR_x**: Rising trigger event configuration bit of line x

- 0: Rising trigger disabled (for Event and Interrupt) for input line
- 1: Rising trigger enabled (for Event and Interrupt) for input line

Note: *The external wakeup lines are edge triggered, no glitch must be generated on these lines. If a rising edge occurs on the external interrupt line while writing to the EXTI_RTSR register, the pending bit is set.*

Rising and falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.

9.3.4 Falling trigger selection register (EXTI_FTSR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														TR22	TR21
									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 31:23 Reserved, must be kept at reset value.

Bits 22:0 **TR_x**: Falling trigger event configuration bit of line x

- 0: Falling trigger disabled (for Event and Interrupt) for input line
- 1: Falling trigger enabled (for Event and Interrupt) for input line

Note: *The external wakeup lines are edge triggered, no glitch must be generated on these lines. If a falling edge occurs on the external interrupt line while writing 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 configuration, both generate a trigger condition.

9.3.5 Software interrupt event register (EXTI_SWIER)

Address offset: 0x10

Reset value: 0x0000 0000

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

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

Bits 22:0 **SWIERx**: Software Interrupt on line x

Writing a 1 to this bit when it is at 0 sets the corresponding pending bit in EXTI_PR. If the interrupt is enabled on this line on the EXTI_IMR and EXTI_EMR, an interrupt request is generated.

This bit is cleared by clearing the corresponding bit in EXTI_PR (by writing a 1 to the bit).

9.3.6 Pending register (EXTI_PR)

Address offset: 0x14

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PR22	PR21	PR20	PR19	PR18	PR17	PR16
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 31:23 Reserved, must be kept at reset value.

Bits 22:0 **PRx**: Pending bit

0: No trigger request occurred

1: selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 to the bit or by changing the sensitivity of the edge detector.

9.3.7 EXTI register map

Table 31 gives the EXTI register map and the reset values.

Table 31. External interrupt/event controller register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	EXTI_IMR	Reserved	MR[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	0	0			
0x04	EXTI_EMR	Reserved	MR[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	0	0			
0x08	EXTI_RTSR	Reserved	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	0	0			
0x0C	EXTI_FTSR	Reserved	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	0	0			
0x10	EXTI_SWIER	Reserved	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	0	0			
0x14	EXTI_PR	Reserved	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	0	0			

Refer to *Table 1 on page 50* for the register boundary addresses.

10 Analog-to-digital converter (ADC)

This section applies to the whole STM32F40x and STM32F41x family, unless otherwise specified.

10.1 ADC introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 19 multiplexed channels allowing it to measure signals from 16 external sources, two internal sources, and the V_{BAT} channel. The A/D conversion of the channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored into a left- or right-aligned 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage goes beyond the user-defined, higher or lower thresholds.

10.2 ADC main features

- 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
- Interrupt generation at the end of conversion, end of injected conversion, and in case of analog watchdog or overrun events
- Single and continuous conversion modes
- Scan mode for automatic conversion of channel 0 to channel ‘n’
- Data alignment with in-built data coherency
- Channel-wise programmable sampling time
- External trigger option with configurable polarity for both regular and injected conversions
- Discontinuous mode
- Dual/Triple mode (on devices with 2 ADCs or more)
- Configurable DMA data storage in Dual/Triple ADC mode
- Configurable delay between conversions in Dual/Triple interleaved mode
- ADC conversion type (refer to the datasheets)
- ADC supply requirements: 2.4 V to 3.6 V at full speed and down to 1.8 V at slower speed
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$
- DMA request generation during regular channel conversion

Figure 28 shows the block diagram of the ADC.

Note: V_{REF-} , if available (depending on package), must be tied to V_{SSA} .

10.3 ADC functional description

Figure 28 shows a single ADC block diagram and *Table 32* gives the ADC pin description.

Figure 28. Single ADC block diagram

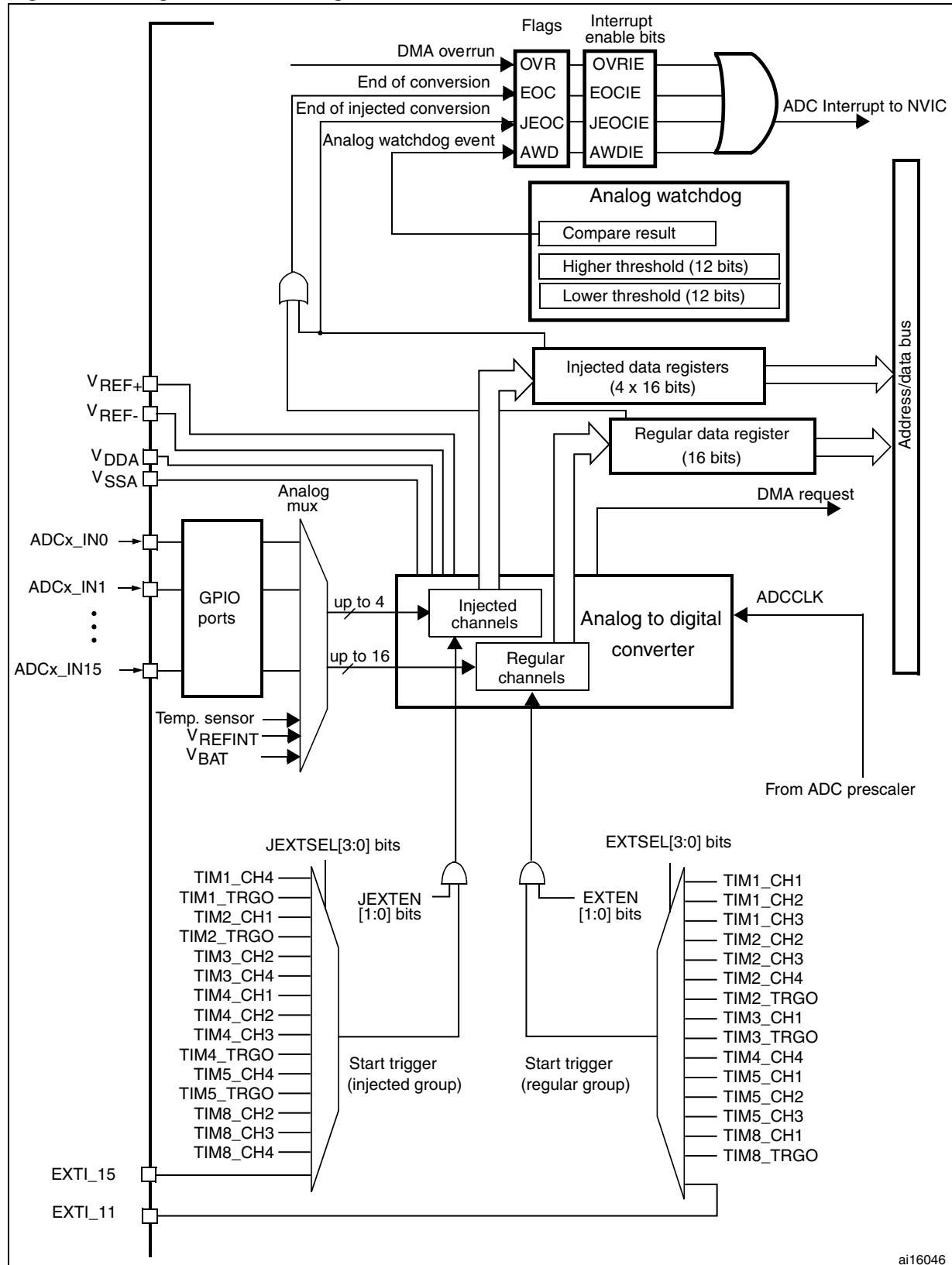


Table 32. ADC pins

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the ADC, $1.8 \text{ V} \leq V_{REF+} \leq V_{DDA}$
V_{DDA}	Input, analog supply	Analog power supply equal to V_{DD} and $2.4 \text{ V} \leq V_{DDA} \leq V_{DD}$ (3.6 V) for full speed $1.8 \text{ V} \leq V_{DDA} \leq V_{DD}$ (3.6 V) for reduced speed
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}
ADCx_IN[15:0]	Analog input signals	16 analog input channels

10.3.1 ADC on-off control

The ADC is powered on by setting the ADON bit in the ADC_CR2 register. When the ADON bit is set for the first time, it wakes up the ADC from the Power-down mode.

Conversion starts when either the SWSTART or the JSWSTART bit is set.

You can stop conversion and put the ADC in power down mode by clearing the ADON bit. In this mode the ADC consumes almost no power (only a few μA).

10.3.2 ADC clock

The ADC features two clock schemes:

- Clock for the analog circuitry: ADCCLK, common to all ADCs

This clock is generated from the APB2 clock divided by a programmable prescaler that allows the ADC to work at $f_{PCLK2}/2$, /4, /6 or /8. Refer to the datasheets for the maximum value of ADCCLK.
- Clock for the digital interface (used for registers read/write access)

This clock is equal to the APB2 clock. The digital interface clock can be enabled/disabled individually for each ADC through the RCC APB2 peripheral clock enable register (RCC_APB2ENR).

10.3.3 Channel selection

There are 16 multiplexed channels. It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions 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 ADC_SQRx registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.
- 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 ADC_JSQR register.

The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

If the ADC_SQRx or ADC_JSQR registers are modified during a conversion, the current conversion is reset and a new start pulse is sent to the ADC to convert the newly chosen group.

Temperature sensor, V_{REFINT} and V_{BAT} internal channels

The temperature sensor is connected to channel ADC1_IN16 and the internal reference voltage V_{REFINT} is connected to ADC1_IN17. These two internal channels can be selected and converted as injected or regular channels.

The V_{BAT} channel is connected to channel ADC1_IN18. It can also be converted as an injected or regular channel.

Note: *The temperature sensor, V_{REFINT} and the V_{BAT} channel are available only on the master ADC1 peripheral.*

10.3.4 Single conversion mode

In Single conversion mode the ADC does one conversion. This mode is started with the CONT bit at 0 by either:

- setting the SWSTART bit in the ADC_CR2 register (for a regular channel only)
- setting the JSWSTART bit (for an injected channel)
- external trigger (for a regular or injected channel)

Once the conversion of the selected channel is complete:

- If a regular channel was converted:
 - The converted data are stored into the 16-bit ADC_DR register
 - The EOC (end of conversion) flag is set
 - An interrupt is generated if the EOCIE bit is set
- If an injected channel was converted:
 - The converted data are stored into the 16-bit ADC_JDR1 register
 - The JEOC (end of conversion injected) flag is set
 - An interrupt is generated if the JEOCIE bit is set

Then the ADC stops.

10.3.5 Continuous conversion mode

In continuous conversion mode, the ADC starts a new conversion as soon as it finishes one. This mode is started with the CONT bit at 1 either by external trigger or by setting the SWSTRT bit in the ADC_CR2 register (for regular channels only).

After each conversion:

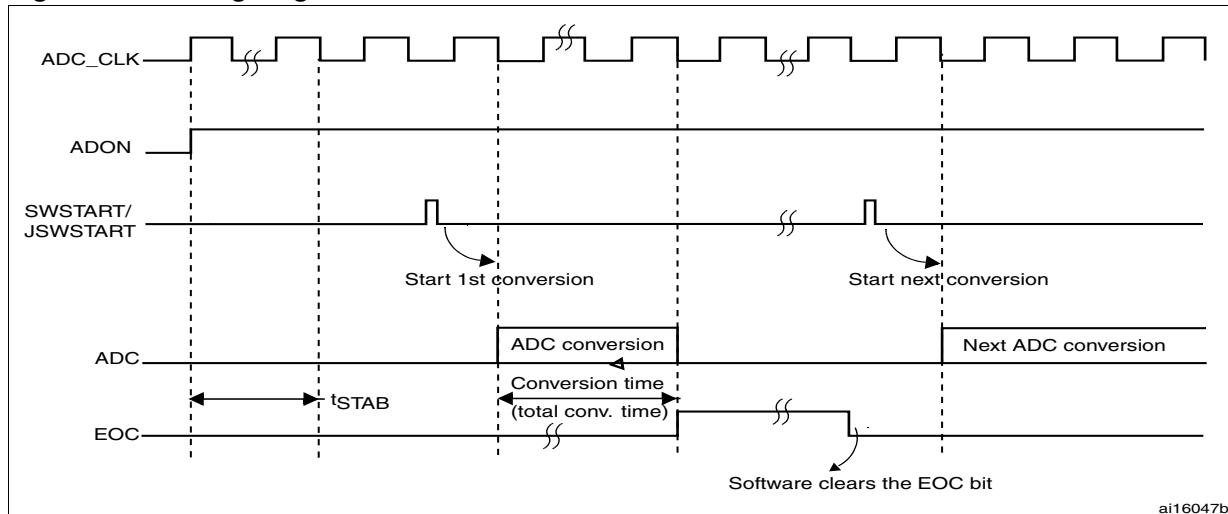
- If a regular group of channels was converted:
 - The last converted data are stored into the 16-bit ADC_DR register
 - The EOC (end of conversion) flag is set
 - An interrupt is generated if the EOCIE bit is set

Note: 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](#) section).

10.3.6 Timing diagram

As shown in [Figure 29](#), the ADC needs a stabilization time of t_{STAB} before it starts converting accurately. After the start of the ADC conversion and after 15 clock cycles, the EOC flag is set and the 16-bit ADC data register contains the result of the conversion.

Figure 29. Timing diagram



10.3.7 Analog watchdog

The AWD analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in the 12 least significant bits of the ADC_HTR and ADC_LTR 16-bit registers. An interrupt can be enabled by using the AWDIE bit in the ADC_CR1 register.

The threshold value is independent of the alignment selected by the ALIGN bit in the ADC_CR2 register. The analog voltage is compared to the lower and higher thresholds before alignment.

[Table 33](#) shows how the ADC_CR1 register should be configured to enable the analog watchdog on one or more channels.

Figure 30. Analog watchdog's guarded area

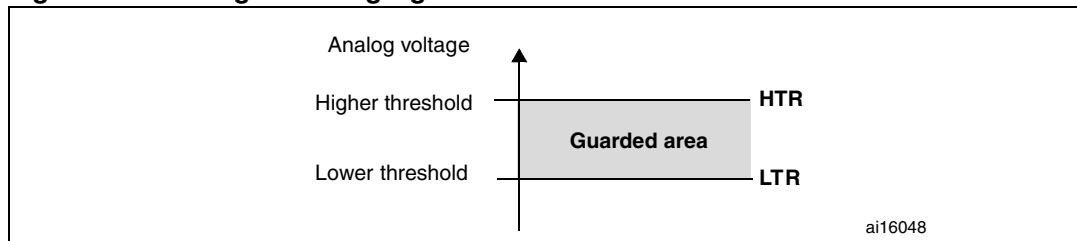


Table 33. Analog watchdog channel selection

Channels guarded by the analog watchdog	ADC_CR1 register control bits (x = don't care)		
	AWDSGL bit	AWDEN bit	JAWDEN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single ⁽¹⁾ injected channel	1	0	1
Single ⁽¹⁾ regular channel	1	1	0
Single ⁽¹⁾ regular or injected channel	1	1	1

1. Selected by the AWDCH[4:0] bits

10.3.8 Scan mode

This mode is used to scan a group of analog channels.

The Scan mode is selected by setting the SCAN bit in the ADC_CR1 register. Once this bit has been set, the ADC scans all the channels selected in the ADC_SQRx registers (for regular channels) or in the ADC_JSQR register (for injected channels). A single conversion is performed for each channel of the group. After each end of conversion, the next channel in the group is converted automatically. If the CONT bit is set, regular channel conversion does not stop at the last selected channel in the group but continues again from the first selected channel.

If the DMA bit is set, the direct memory access (DMA) controller is used to transfer the data converted from the regular group of channels (stored in the ADC_DR register) to SRAM after each regular channel conversion.

The EOC bit is set in the ADC_SR register:

- At the end of each regular group sequence if the EOCS bit is cleared to 0
- At the end of each regular channel conversion if the EOCS bit is set to 1

The data converted from an injected channel are always stored into the ADC_JDRx registers.

10.3.9 Injected channel management

Triggered injection

To use triggered injection, the JAUTO bit must be cleared in the ADC_CR1 register.

1. Start the conversion of a group of regular channels either by external trigger or by setting the SWSTART bit in the ADC_CR2 register.
 2. If an external injected trigger occurs or if the JSWSTART bit is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches to Scan-once mode.
 3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
- 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 31](#) 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 30 ADC clock cycles (that is two conversions with a sampling time of 3 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.

Auto-injection

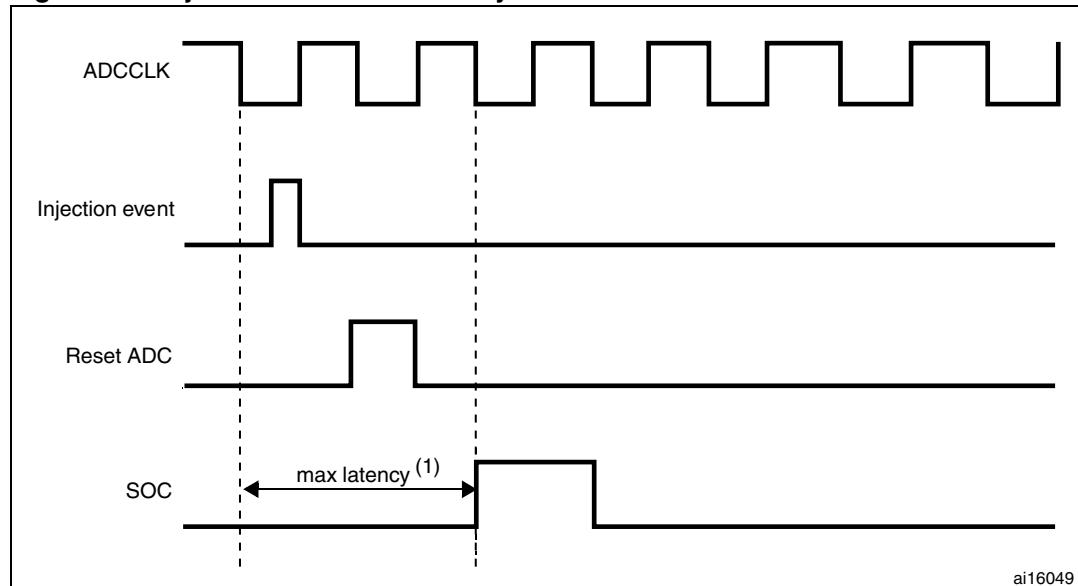
If the JAUTO bit is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRx and ADC_JSQR registers.

In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.

Figure 31. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the STM32F40x and STM32F41x datasheets.

10.3.10 Discontinuous mode

Regular group

This mode is enabled by setting the DISCEN bit in the ADC_CR1 register. It can be used to convert a short sequence of n conversions ($n \leq 8$) that is part of the sequence of conversions selected in the ADC_SQRx registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRx registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

n = 3, channels to be converted = 0, 1, 2, 3, 6, 7, 9, 10
1st trigger: sequence converted 0, 1, 2
2nd trigger: sequence converted 3, 6, 7
3rd trigger: sequence converted 9, 10 and an EOC event generated
4th trigger: sequence converted 0, 1, 2

Note: When a regular group is converted in discontinuous mode, no rollover occurs.

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 0, 1 and 2 in the 1st subgroup.

Injected group

This mode is enabled by setting the JDISCEN bit in the ADC_CR1 register. It can be used to convert the sequence selected in the ADC_JSQR register, channel by channel, after an external trigger event.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:

n = 1, channels to be converted = 1, 2, 3
1st trigger: channel 1 converted
2nd trigger: channel 2 converted
3rd trigger: channel 3 converted and EOC and JEOC events generated
4th trigger: channel 1

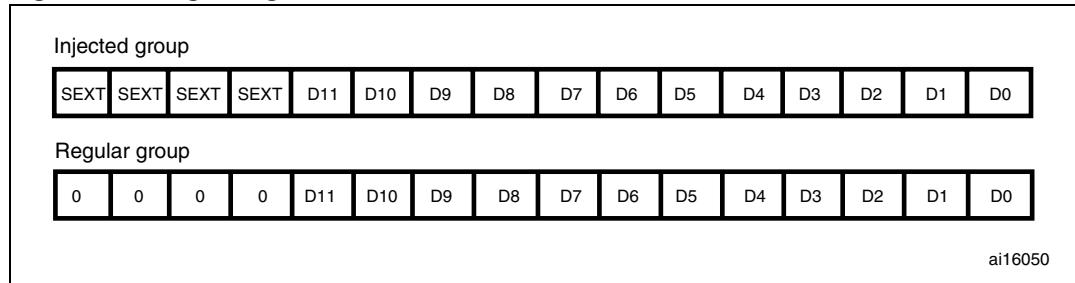
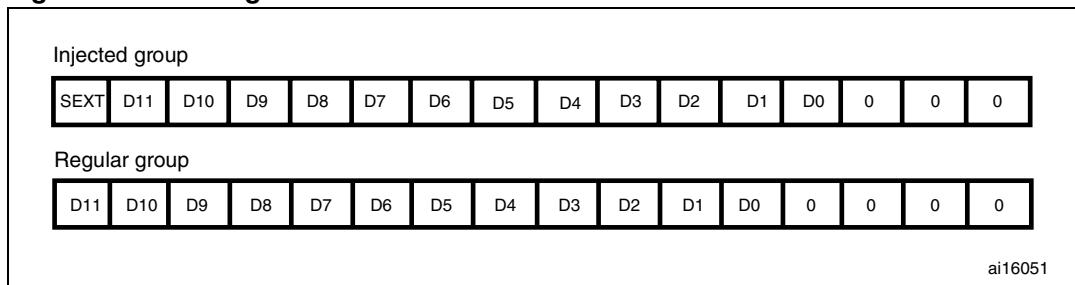
- Note:
- 1 When all injected channels are converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.
 - 2 It is not possible to use both the auto-injected and discontinuous modes simultaneously.
 - 3 Discontinuous mode must not be set for regular and injected groups at the same time.
Discontinuous mode must be enabled only for the conversion of one group.

10.4 Data alignment

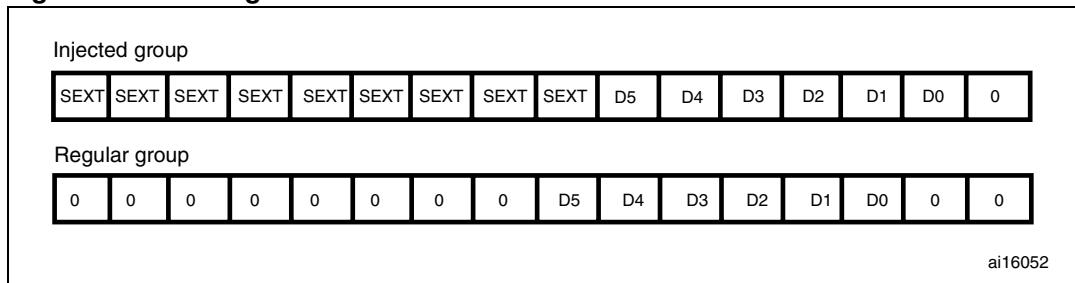
The ALIGN bit in the ADC_CR2 register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in [Figure 32](#) and [Figure 33](#).

The converted data value from the injected group of channels is decreased by the user-defined offset written in the ADC_JOFRx registers so the result can be a negative value. The SEXT bit represents the extended sign value.

For channels in a regular group, no offset is subtracted so only twelve bits are significant.

Figure 32. Right alignment of 12-bit data**Figure 33. Left alignment of 12-bit data**

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 34](#).

Figure 34. Left alignment of 6-bit data

10.5 Channel-wise programmable sampling time

The ADC samples the input voltage for a number of ADCCLK cycles that can be modified using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. Each channel can be sampled with a different sampling time.

The total conversion time is calculated as follows:

$$T_{\text{conv}} = \text{Sampling time} + 12 \text{ cycles}$$

Example:

With ADCCLK = 38 MHz and sampling time = 3 cycles:

$$T_{\text{conv}} = 3 + 12 = 15 \text{ cycles} = 0.5 \mu\text{s} \text{ with APB2 at } 60 \text{ MHz}$$

10.6 Conversion on external trigger and trigger polarity

Conversion can be triggered by an external event (e.g. timer capture, EXTI line). 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. [Table 34](#) provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 34. Configuring the trigger polarity

Source	EXTEN[1:0] / JEXTEN[1:0]
Trigger detection disabled	00
Detection on the rising edge	01
Detection on the falling edge	10
Detection on both the rising and falling edges	11

Note: The polarity of the external trigger can be changed on the fly.

The EXTSEL[3:0] and JEXTSEL[3:0] control bits are used to select which out of 16 possible events can trigger conversion for the regular and injected groups.

[Table 35](#) gives the possible external trigger for regular conversion.

Table 35. External trigger for regular channels

Source	Type	EXTSEL[3:0]
TIM1_CH1 event	Internal signal from on-chip timers	0000
TIM1_CH2 event		0001
TIM1_CH3 event		0010
TIM2_CH2 event		0011
TIM2_CH3 event		0100
TIM2_CH4 event		0101
TIM2_TRGO event		0110
TIM3_CH1 event		0111
TIM3_TRGO event		1000
TIM4_CH4 event		1001
TIM5_CH1 event		1010
TIM5_CH2 event		1011
TIM5_CH3 event		1100
TIM8_CH1 event		1101
TIM8_TRGO event		1110
EXTI line11	External pin	1111

[Table 36](#) gives the possible external trigger for injected conversion.

Table 36. External trigger for injected channels

Source	Connection type	JEXTSEL[3:0]
TIM1_CH4 event	Internal signal from on-chip timers	0000
TIM1_TRGO event		0001
TIM2_CH1 event		0010
TIM2_TRGO event		0011
TIM3_CH2 event		0100
TIM3_CH4 event		0101
TIM4_CH1 event		0110
TIM4_CH2 event		0111
TIM4_CH3 event		1000
TIM4_TRGO event		1001
TIM5_CH4 event		1010
TIM5_TRGO event		1011
TIM8_CH2 event		1100
TIM8_CH3 event		1101
TIM8_CH4 event		1110
EXTI line15	External pin	1111

Software source trigger events can be generated by setting SWSTART (for regular conversion) or JSWSTART (for injected conversion) in ADC_CR2.

A regular group conversion can be interrupted by an injected trigger.

Note:

The trigger selection can be changed on the fly. However, when the selection changes, there is a time frame of 1 APB clock cycle during which the trigger detection is disabled. This is to avoid spurious detection during transitions.

10.7 Fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution. The RES bits are used to select the number of bits available in the data register. The minimum conversion time for each resolution is then as follows:

- 12 bits: $3 + 12 = 15$ ADCCLK cycles
- 10 bits: $3 + 10 = 13$ ADCCLK cycles
- 8 bits: $3 + 8 = 11$ ADCCLK cycles
- 6 bits: $3 + 6 = 9$ ADCCLK cycles

10.8 Data management

10.8.1 Using the DMA

Since converted regular channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one regular channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMA bit set to 1 in the ADC_CR2 register), after each conversion of a regular channel, a DMA request is generated. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Despite this, if data are lost (overrun), the OVR bit in the ADC_SR register is set and an interrupt is generated (if the OVRIE enable bit is set). DMA transfers are then disabled and DMA requests are no longer accepted. In this case, if a DMA request is made, the regular conversion in progress is aborted and further regular triggers are ignored. It is then necessary to clear the OVR flag and the DMAEN bit in the used DMA stream, and to re-initialize both the DMA and the ADC to have the wanted converted channel data transferred to the right memory location. Only then can the conversion be resumed and the data transfer, enabled again. Injected channel conversions are not impacted by overrun errors.

When OVR = 1 in DMA mode, the DMA requests are blocked after the last valid data have been transferred, which means that all the data transferred to the RAM can be considered as valid.

At the end of the last DMA transfer (number of transfers configured in the DMA controller's DMA_SxRTR register):

- No new DMA request is issued to the DMA controller if the DDS bit is cleared to 0 in the ADC_CR2 register (this avoids generating an overrun error). However the DMA bit is not cleared by hardware. It must be written to 0, then to 1 to start a new transfer.
- Requests can continue to be generated if the DDS bit is set to 1. This allows configuring the DMA in double-buffer circular mode.

10.8.2 Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the EOCS bit must be set in the ADC_CR2 register for the EOC status bit to be set at the end of each conversion, and not only at the end of the sequence. When EOCS = 1, overrun detection is automatically enabled. Thus, each time a conversion is complete, EOC is set and the ADC_DR register can be read. The overrun management is the same as when the DMA is used.

10.8.3 Conversions without DMA and without overrun detection

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). For that, the DMA must be disabled (DMA = 0) and the EOC bit must be set at the end of a sequence only (EOCS = 0). In this configuration, overrun detection is disabled.

10.9 Multi ADC mode

In devices with two ADCs or more, the Dual (with two ADCs) and Triple (with three ADCs) ADC modes can be used (see [Figure 35](#)).

In multi ADC mode, the start of conversion is triggered alternately or simultaneously by the ADC1 master to the ADC2 and ADC3 slaves, depending on the mode selected by the MULTI[4:0] bits in the ADC_CCR register.

Note: *In multi ADC mode, when configuring conversion trigger by an external event, the application must set trigger by the master only and disable trigger by slaves to prevent spurious triggers that would start unwanted slave conversions.*

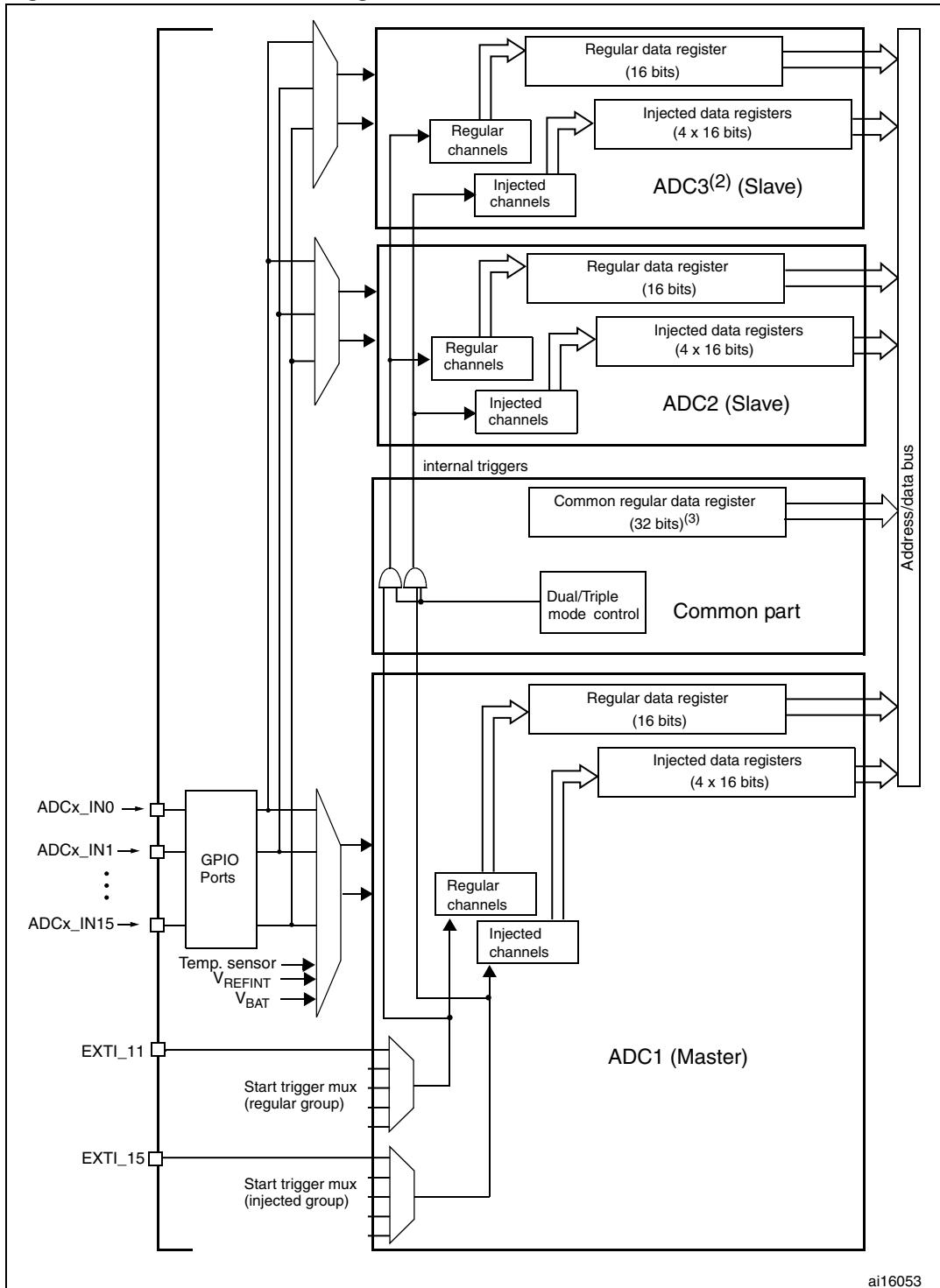
The four possible modes below are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use the previous modes combined in the following ways:

- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode

Note: *In multi ADC mode, the converted data can be read on the multi-mode data register (ADC_CDR). The status bits can be read in the multi-mode status register (ADC_CSR).*

Figure 35. Multi ADC block diagram⁽¹⁾

1. Although external triggers are present on ADC2 and ADC3 they are not shown in this diagram.
2. In the Dual ADC mode, the ADC3 slave part is not present.
3. In Triple ADC mode, the ADC common data register (ADC_CDR) contains the ADC1, ADC2 and ADC3's regular converted data. All 32 register bits are used according to a selected storage order.
In Dual ADC mode, the ADC common data register (ADC_CDR) contains both the ADC1 and ADC2's regular converted data. All 32 register bits are used.

- DMA requests in Multi ADC mode:

In Multi ADC mode the DMA may be configured to transfer converted data in three different modes. In all cases, the DMA streams to use are those connected to the ADC:

- **DMA mode 1:** On each DMA request (one data item is available), a half-word representing an ADC-converted data item is transferred.

In Dual ADC mode, ADC1 data are transferred on the first request, ADC2 data are transferred on the second request and so on.

In Triple ADC mode, ADC1 data are transferred on the first request, ADC2 data are transferred on the second request and ADC3 data are transferred on the third request; the sequence is repeated. So the DMA first transfers ADC1 data followed by ADC2 data followed by ADC3 data and so on.

DMA mode 1 is used in regular simultaneous triple mode.

Example:

Regular simultaneous triple mode: 3 consecutive DMA requests are generated (one for each converted data item)

1st request: $\text{ADC_CDR}[31:0] = \text{ADC1_DR}[15:0]$

2nd request: $\text{ADC_CDR}[31:0] = \text{ADC2_DR}[15:0]$

3rd request: $\text{ADC_CDR}[31:0] = \text{ADC3_DR}[15:0]$

4th request: $\text{ADC_CDR}[31:0] = \text{ADC1_DR}[15:0]$

- **DMA mode 2:** On each DMA request (two data items are available) two half-words representing two ADC-converted data items are transferred as a word.

In Dual ADC mode, both ADC2 and ADC1 data are transferred on the first request (ADC2 data take the upper half-word and ADC1 data take the lower half-word) and so on.

In Triple ADC mode, three DMA requests are generated. On the first request, both ADC2 and ADC1 data are transferred (ADC2 data take the upper half-word and ADC1 data take the lower half-word). On the second request, both ADC1 and ADC3 data are transferred (ADC1 data take the upper half-word and ADC3 data take the lower half-word). On the third request, both ADC3 and ADC2 data are transferred (ADC3 data take the upper half-word and ADC2 data take the lower half-word) and so on.

DAM mode 2 is used in interleaved mode and in regular simultaneous mode (for Dual ADC mode only).

Example:

- a) Interleaved dual mode: a DMA request is generated each time 2 data items are available:

1st request: $\text{ADC_CDR}[31:0] = \text{ADC2_DR}[15:0] \mid \text{ADC1_DR}[15:0]$

2nd request: $\text{ADC_CDR}[31:0] = \text{ADC2_DR}[15:0] \mid \text{ADC1_DR}[15:0]$

- b) Interleaved triple mode: a DMA request is generated each time 2 data items are available

1st request: $\text{ADC_CDR}[31:0] = \text{ADC2_DR}[15:0] \mid \text{ADC1_DR}[15:0]$

2nd request: $\text{ADC_CDR}[31:0] = \text{ADC1_DR}[15:0] \mid \text{ADC3_DR}[15:0]$

3rd request: $\text{ADC_CDR}[31:0] = \text{ADC3_DR}[15:0] \mid \text{ADC2_DR}[15:0]$

4th request: $\text{ADC_CDR}[31:0] = \text{ADC2_DR}[15:0] \mid \text{ADC1_DR}[15:0]$

- **DMA mode 3:** This mode is similar to the DMA mode 2. 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. The data transfer order is similar to that of the DMA mode 2.

DMA mode 3 is used in interleaved mode in 6-bit and 8-bit resolutions.

Example:

- a) Interleaved dual mode: a DMA request is generated each time 2 data items are available
 1st request: $\text{ADC_CDR}[15:0] = \text{ADC2_DR}[7:0] \mid \text{ADC1_DR}[7:0]$
 2nd request: $\text{ADC_CDR}[15:0] = \text{ADC2_DR}[7:0] \mid \text{ADC1_DR}[7:0]$
- b) Interleaved triple mode: a DMA request is generated each time 2 data items are available
 1st request: $\text{ADC_CDR}[15:0] = \text{ADC2_DR}[7:0] \mid \text{ADC1_DR}[7:0]$
 2nd request: $\text{ADC_CDR}[15:0] = \text{ADC1_DR}[7:0] \mid \text{ADC3_DR}[15:0]$
 3rd request: $\text{ADC_CDR}[15:0] = \text{ADC3_DR}[7:0] \mid \text{ADC2_DR}[7:0]$
 4th request: $\text{ADC_CDR}[15:0] = \text{ADC2_DR}[7:0] \mid \text{ADC1_DR}[7:0]$

Overrun detection: If an overrun is detected on one of the concerned ADCs (ADC1 and ADC2 in dual and triple modes, ADC3 in triple mode only), the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid. It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

10.9.1 Injected simultaneous mode

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of ADC1 (selected by the JEXTSEL[3:0] bits in the ADC1_CR2 register). A simultaneous trigger is provided to ADC2 and ADC3.

Note: *Do not convert the same channel on the two/three ADCs (no overlapping sampling times for the two/three ADCs when converting the same channel).*

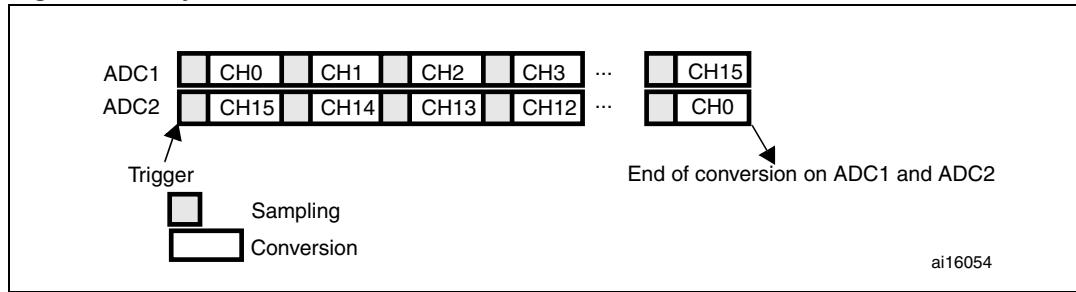
In simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer of the 2 sequences (Dual ADC mode) /3 sequences (Triple ADC mode). Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence 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.

Dual ADC mode

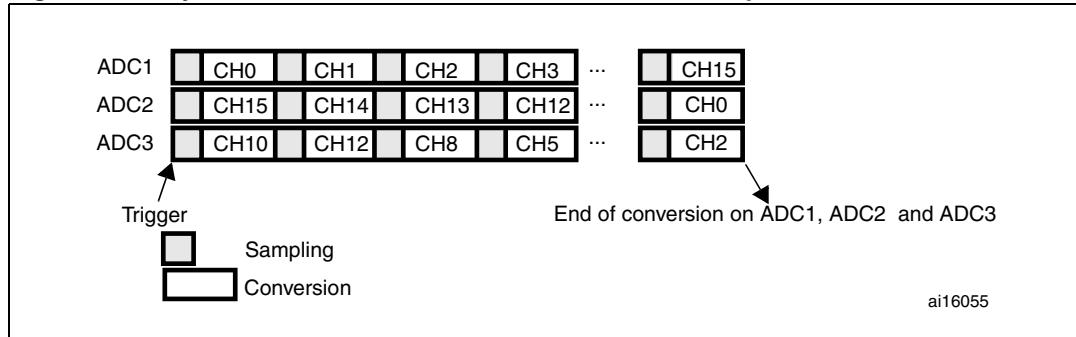
At the end of conversion event on ADC1 or ADC2:

- The converted data are stored into the ADC_JDRx registers of each ADC interface.
- A JEOC interrupt is generated (if enabled on one of the two ADC interfaces) when the ADC1/ADC2's injected channels have all been converted.

Figure 36. Injected simultaneous mode on 4 channels: dual ADC mode**Triple ADC mode**

At the end of conversion event on ADC1, ADC2 or ADC3:

- The converted data are stored into the ADC_JDRx registers of each ADC interface.
- A JEOC interrupt is generated (if enabled on one of the three ADC interfaces) when the ADC1/ADC2/ADC3's injected channels have all been converted.

Figure 37. Injected simultaneous mode on 4 channels: triple ADC mode**10.9.2 Regular simultaneous mode**

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of ADC1 (selected by the EXTSEL[3:0] bits in the ADC1_CR2 register). A simultaneous trigger is provided to ADC2 and ADC3.

Note: *Do not convert the same channel on the two/three ADCs (no overlapping sampling times for the two/three 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 long conversion time of the 2 sequences (Dual ADC mode) /3 sequences (Triple ADC mode). Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

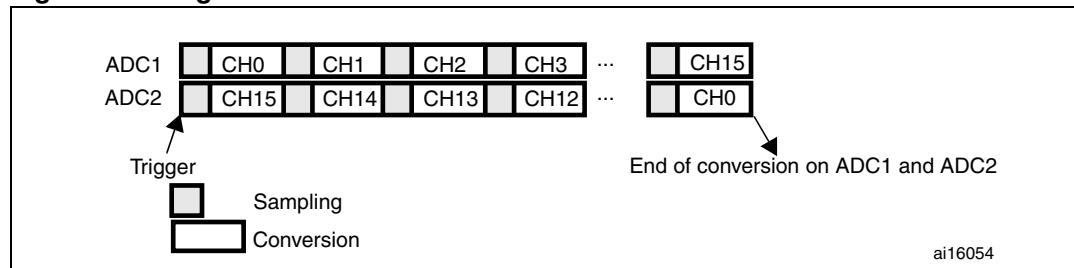
Injected conversions must be disabled.

Dual ADC mode

At the end of conversion event on ADC1 or ADC2:

- A 32-bit DMA transfer request is generated (if DMA[1:0] bits in the ADC_CCR register are equal to 0b10). This request transfers the ADC2 converted data stored in the upper half-word of the ADC_CDR 32-bit register to the SRAM and then the ADC1 converted data stored in the lower half-word of ADC_CCR to the SRAM.
- An EOC interrupt is generated (if enabled on one of the two ADC interfaces) when the ADC1/ADC2's regular channels have all been converted.

Figure 38. Regular simultaneous mode on 16 channels: dual ADC mode

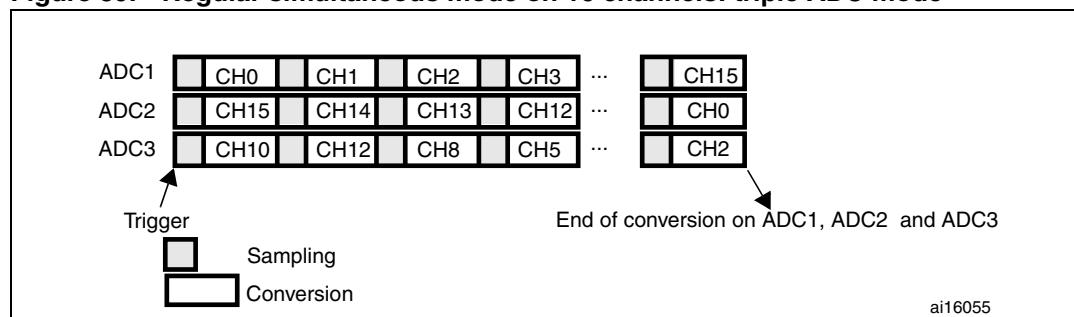


Triple ADC mode

At the end of conversion event on ADC1, ADC2 or ADC3:

- Three 32-bit DMA transfer requests are generated (if DMA[1:0] bits in the ADC_CCR register are equal to 0b01). Three transfers then take place from the ADC_CDR 32-bit register to SRAM: first the ADC1 converted data, then the ADC2 converted data and finally the ADC3 converted data. The process is repeated for each new three conversions.
- An EOC interrupt is generated (if enabled on one of the three ADC interfaces) when the ADC1/ADC2/ADC3's regular channels have all been converted.

Figure 39. Regular simultaneous mode on 16 channels: triple ADC mode



10.9.3 Interleaved mode

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of ADC1.

Dual ADC mode

After an external trigger occurs:

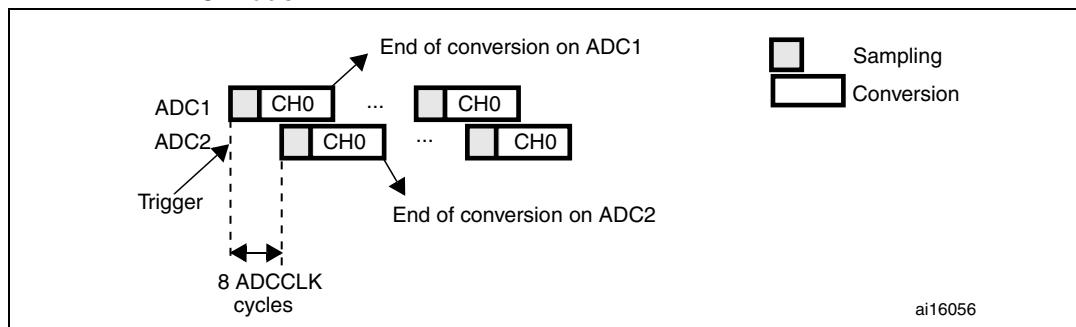
- ADC1 starts immediately
- ADC2 starts after a delay of several-ADC clock cycles

The minimum delay which separates 2 conversions in interleaved mode is configured in the DELAY bits in the ADC_CCR register. However, 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). In this case, the delay becomes the sampling time + 2 ADC clock cycles. For instance, if $\text{DELAY} = 5$ clock cycles and the sampling takes 15 clock cycles on both ADCs, then 17 clock cycles will separate conversions on ADC1 and ADC2).

If the CONT bit is set on both ADC1 and ADC2, the selected regular channels of both ADCs are continuously converted.

After an EOC interrupt is generated by ADC2 (if enabled through the EOCIE bit) a 32-bit DMA transfer request is generated (if the DMA[1:0] bits in ADC_CCR are equal to 0b10). This request first transfers the ADC2 converted data stored in the upper half-word of the ADC_CDR 32-bit register into SRAM, then the ADC1 converted data stored in the register's lower half-word into SRAM.

Figure 40. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode



Triple ADC mode

After an external trigger occurs:

- ADC1 starts immediately and
- ADC2 starts after a delay of several ADC clock cycles
- ADC3 starts after a delay of several ADC clock cycles referred to the ADC2 conversion

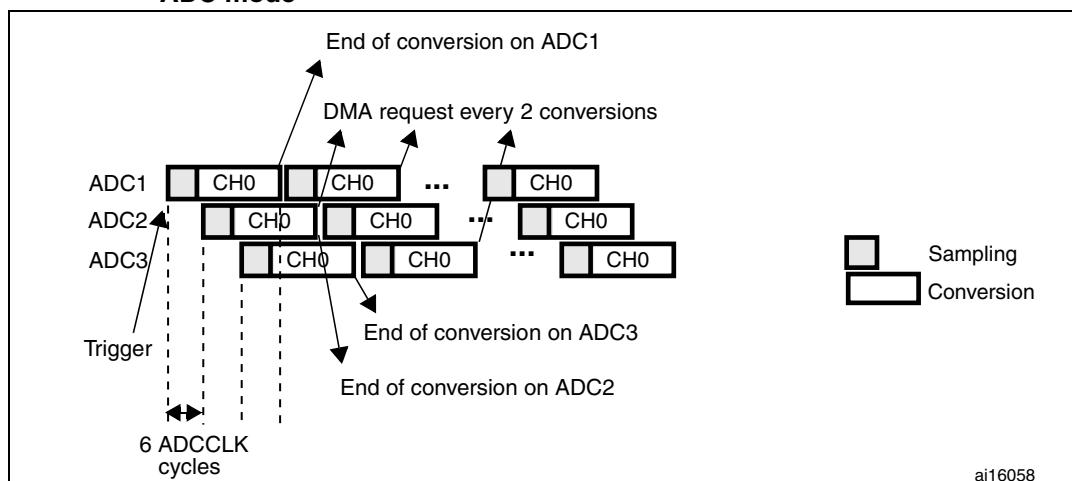
The minimum delay which separates 2 conversions in interleaved mode is configured in the DELAY bits in the ADC_CCR register. However, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time). In this case, the delay becomes the sampling time + 2 ADC clock cycles. For instance, if $\text{DELAY} = 5$ clock cycles and the sampling takes 15 clock cycles on the three ADCs, then 17 clock cycles will separate the conversions on ADC1, ADC2 and ADC3).

If the CONT bit is set on ADC1, ADC2 and ADC3, the selected regular channels of all ADCs are continuously converted.

In this mode a DMA request is generated each time 2 data items are available, (if the DMA[1:0] bits in the ADC_CCR register are equal to 0b10). The request first transfers the first converted data stored in the lower half-word of the ADC_CDR 32-bit register to SRAM, then it transfers the second converted data stored in ADC_CDR's upper half-word to SRAM. The sequence is the following:

- 1st request: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]
- 2nd request: ADC_CDR[31:0] = ADC1_DR[15:0] | ADC3_DR[15:0]
- 3rd request: ADC_CDR[31:0] = ADC3_DR[15:0] | ADC2_DR[15:0]
- 4th request: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0], ...

Figure 41. Interleaved mode on 1 channel in continuous conversion mode: triple ADC mode



10.9.4 Alternate trigger mode

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of ADC1.

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.

Dual ADC mode

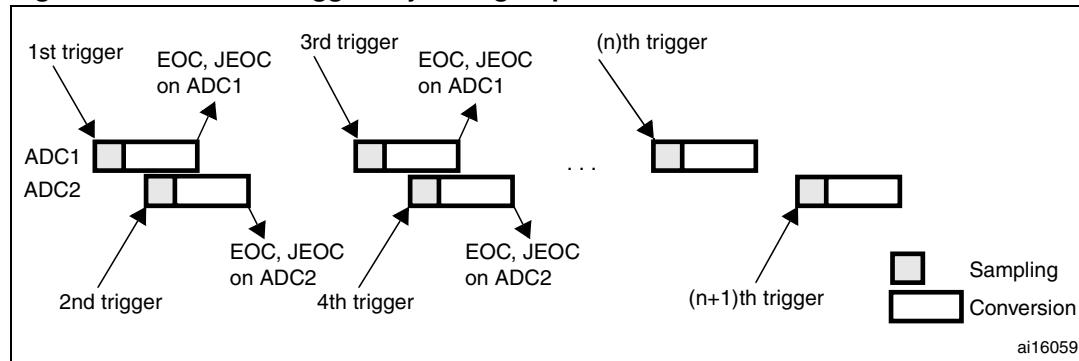
- When the 1st trigger occurs, all injected ADC1 channels in the group are converted
- When the 2nd trigger occurs, all injected ADC2 channels in the group are converted
- and so on

A JEOC interrupt, if enabled, is generated after all injected ADC1 channels in the group have been converted.

A JEOC interrupt, if enabled, is generated after all injected ADC2 channels in the group have been converted.

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 ADC1 channels in the group.

Figure 42. Alternate trigger: injected group of each ADC



If the injected discontinuous mode is enabled for both ADC1 and ADC2:

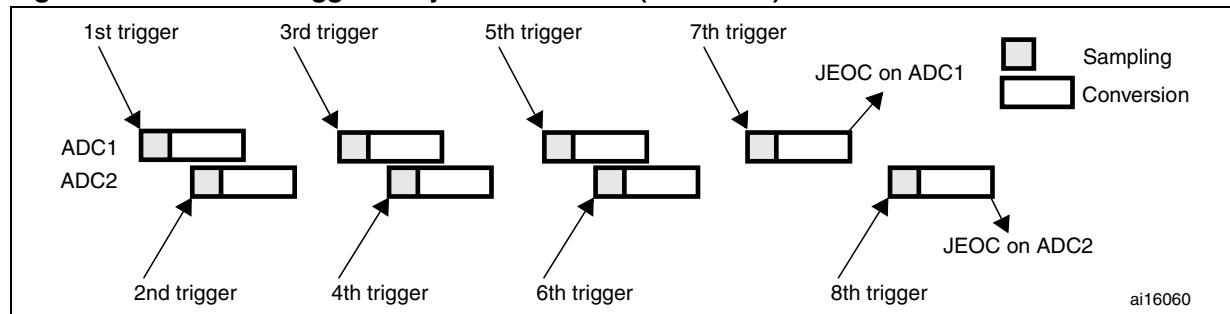
- When the 1st trigger occurs, the first injected ADC1 channel is converted.
- When the 2nd trigger occurs, the first injected ADC2 channel are converted
- and so on

A JEOC interrupt, if enabled, is generated after all injected ADC1 channels in the group have been converted.

A JEOC interrupt, if enabled, is generated after all injected ADC2 channels in the group have been converted.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

Figure 43. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Triple ADC mode

- When the 1st trigger occurs, all injected ADC1 channels in the group are converted.
- When the 2nd trigger occurs, all injected ADC2 channels in the group are converted.
- When the 3rd trigger occurs, all injected ADC3 channels in the group are converted.
- and so on

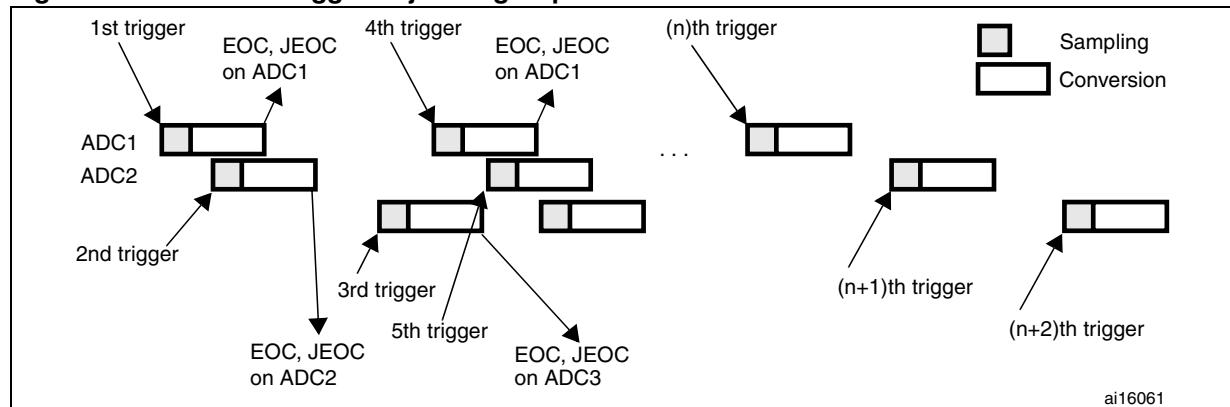
A JEOC interrupt, if enabled, is generated after all injected ADC1 channels in the group have been converted.

A JEOC interrupt, if enabled, is generated after all injected ADC2 channels in the group have been converted.

A JEOC interrupt, if enabled, is generated after all injected ADC3 channels in the group have been converted.

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 ADC1 channels in the group.

Figure 44. Alternate trigger: injected group of each ADC



10.9.5 Combined regular/injected simultaneous mode

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

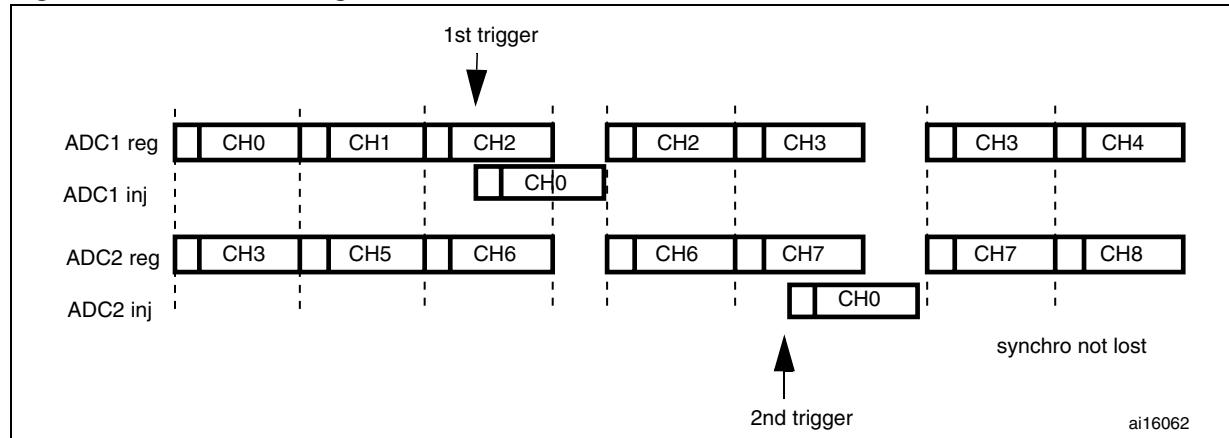
Note: *In combined regular/injected simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences (Dual ADC mode) /3 sequences (Triple ADC mode). Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.*

10.9.6 Combined regular simultaneous + alternate trigger mode

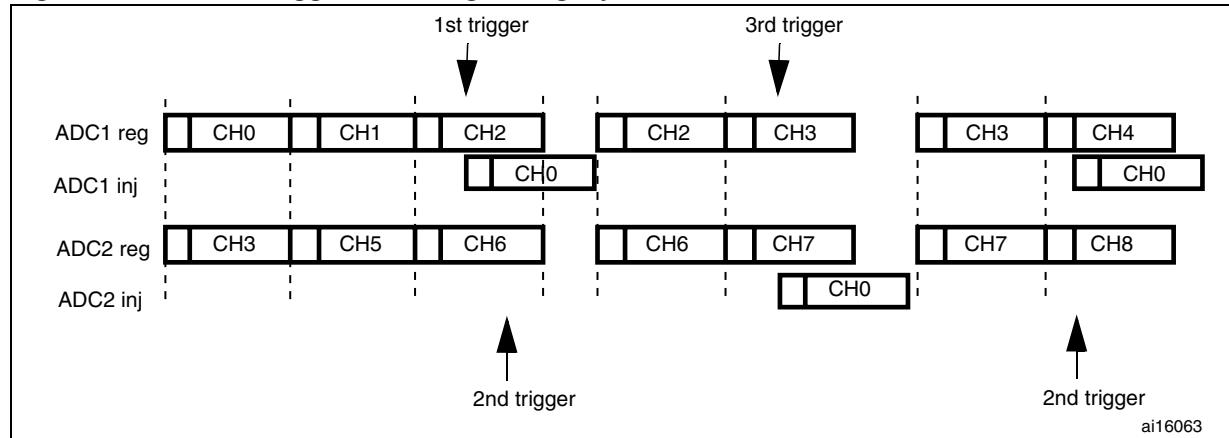
It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. [Figure 45](#) shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If 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 (Dual ADC mode) /3 sequences (Triple ADC mode). Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.*

Figure 45. Alternate + regular simultaneous

If a trigger occurs during an injected conversion that has interrupted a regular conversion, it is ignored. [Figure 46](#) shows the behavior in this case (2nd trigger is ignored).

Figure 46. Case of trigger occurring during injected conversion

10.10 Temperature sensor

The temperature sensor can be used to measure the ambient temperature (T_A) of the device.

The temperature sensor is internally connected to the ADC1_IN16 input channel which is used to convert the sensor's output voltage to a digital value.

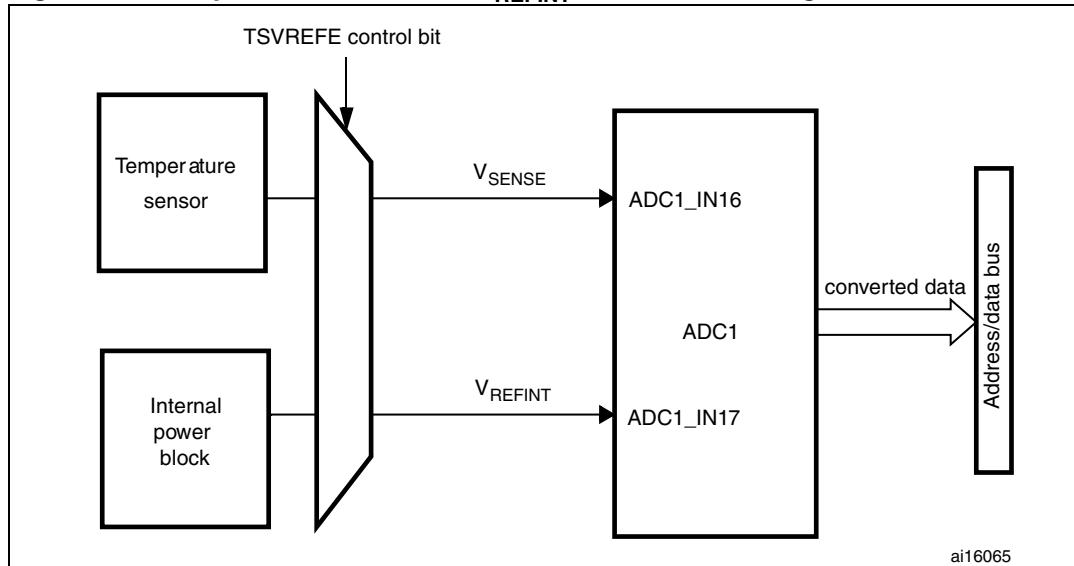
[Figure 47](#) shows the block diagram of the temperature sensor.

When not in use, the sensor can be put in power down mode.

Note: The TSVREFE bit must be set to enable the conversion of both internal channels: ADC1_IN16 (temperature sensor) and ADC1_IN17 (V_{REFINT}).

Main features

- Supported temperature range: -40 to 125 °C
- Precision: ± 1.5 °C

Figure 47. Temperature sensor and V_{REFINT} channel block diagram

Reading the temperature

To use the sensor:

4. Select the ADC1_IN16 input channel
5. Select a sampling time greater than the minimum sampling time specified in the datasheet.
6. Set the TSVREFE bit in the ADC_CCR register to wake up the temperature sensor from power down mode
7. Start the ADC conversion by setting the SWSTART bit (or by external trigger)
8. Read the resulting V_{SENSE} data in the ADC data register
9. Calculate the temperature using the following formula:

$$\text{Temperature (in } ^\circ\text{C)} = \{(V_{\text{SENSE}} - V_{25}) / \text{Avg_Slope}\} + 25$$

Where:

- V₂₅ = V_{SENSE} value for 25° C
- Avg_Slope = average slope of the temperature vs. V_{SENSE} curve (given in mV/°C or µV/°C)

Refer to the datasheet's electrical characteristics section for the actual values of V₂₅ and Avg_Slope.

Note:

The sensor has a startup time after waking from power down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADON and TSVREFE bits should be set at the same time.

10.11 Battery charge monitoring

The VBATE bit in the ADC_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 VBATE is set, to connect V_{BAT}/2 to the ADC1_IN18 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.

10.12 ADC interrupts

An interrupt can be produced on the end of conversion for regular and injected groups, when the analog watchdog status bit is set and when the overrun status bit is set. Separate interrupt enable bits are available for flexibility.

Two other flags are present in the ADC_SR register, but there is no interrupt associated with them:

- JSTART (Start of conversion for channels of an injected group)
- START (Start of conversion for channels of a regular group)

Table 37. ADC interrupts

Interrupt event	Event flag	Enable control bit
End of conversion of a regular group	EOC	EOCIE
End of conversion of an injected group	JEOC	JEOCIE
Analog watchdog status bit is set	AWD	AWDIE
Overrun	OVR	OVRIE

10.13 ADC registers

Refer to [Section 1.1 on page 46](#) for a list of abbreviations used in register descriptions.

10.13.1 ADC status register (ADC_SR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
								OVR	STRT	JSTRT	JEOC	EOC	AWD		
								rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0		

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

Bit 5 **OVR**: Overrun

This bit is set by hardware when data are lost (either in single mode or in dual/triple mode). It is cleared by software. Overrun detection is enabled only when DMA = 1 or EOCS = 1.

- 0: No overrun occurred
- 1: Overrun has occurred

Bit 4 **STRT**: Regular channel start flag

This bit is set by hardware when regular channel conversion starts. It is cleared by software.

- 0: No regular channel conversion started
- 1: Regular channel conversion has started

Bit 3 **JSTRT**: Injected channel start flag

This bit is set by hardware when injected group conversion starts. It is cleared by software.

- 0: No injected group conversion started
- 1: Injected group conversion has started

Bit 2 **JEOC**: Injected channel end of conversion

This bit is set by hardware at the end of the conversion of all injected channels in the group. It is cleared by software.

- 0: Conversion is not complete
- 1: Conversion complete

Bit 1 **EOC**: Regular channel end of conversion

This bit is set by hardware at the end of the conversion of a regular group of channels. It is cleared by software or by reading the ADC_DR register.

- 0: Conversion not complete (EOCS=0), or sequence of conversions not complete (EOCS=1)
- 1: Conversion complete (EOCS=0), or sequence of conversions complete (EOCS=1)

Bit 0 **AWD**: Analog watchdog flag

This bit is set by hardware when the converted voltage crosses the values programmed in the ADC_LTR and ADC_HTR registers. It is cleared by software.

- 0: No analog watchdog event occurred
- 1: Analog watchdog event occurred

10.13.2 ADC control register 1 (ADC_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				OVRIE	RES		AWDEN	JAWDEN	Reserved						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCE N	DISC EN	JAUTO	AWDSG L	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 26 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Bits 25:24 **RES[1:0]**: Resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit (15 ADCCLK cycles)

01: 10-bit (13 ADCCLK cycles)

10: 8-bit (11 ADCCLK cycles)

11: 6-bit (9 ADCCLK cycles)

Bit 23 **AWDEN**: Analog watchdog enable on regular channels

This bit is set and cleared by software.

0: Analog watchdog disabled on regular channels

1: Analog watchdog enabled on regular channels

Bit 22 **JAWDEN**: Analog watchdog enable on injected channels

This bit is set and cleared by software.

0: Analog watchdog disabled on injected channels

1: Analog watchdog enabled on injected channels

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:13 **DISCNUM[2:0]**: Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Bit 12 **JDISCEN**: Discontinuous mode on injected channels

This bit 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

Bit 11 DISCEN: Discontinuous mode on regular channels

This bit is set and cleared by software to enable/disable Discontinuous mode on regular channels.

- 0: Discontinuous mode on regular channels disabled
- 1: Discontinuous mode on regular channels enabled

Bit 10 JAUTO: Automatic injected group conversion

This bit 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

Bit 9 AWDSGL: Enable the watchdog on a single channel in scan mode

This bit is set and cleared by software to enable/disable the analog watchdog on the channel identified by the AWDCH[4:0] bits.

- 0: Analog watchdog enabled on all channels
- 1: Analog watchdog enabled on a single channel

Bit 8 SCAN: Scan mode

This bit is set and cleared by software to enable/disable the Scan mode. In Scan mode, the inputs selected through the ADC_SQRx or ADC_JSQRx registers are converted.

- 0: Scan mode disabled
- 1: Scan mode enabled

Note: An EOC interrupt is generated if the EOCS bit is set:

- At the end of each regular group sequence if the EOCS bit is cleared to 0
- At the end of each regular channel conversion if the EOCS bit is set to 1

Note: A JEOC interrupt is generated only on the end of conversion of the last channel if the JEOCIE bit is set.

Bit 7 JEOCIE: Interrupt enable for injected channels

This bit is set and cleared by software to enable/disable the end of conversion interrupt for injected channels.

- 0: JEOC interrupt disabled
- 1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Bit 6 AWDIE: Analog watchdog interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt. In Scan mode if the watchdog thresholds are crossed, scan is aborted only if this bit is enabled.

- 0: Analog watchdog interrupt disabled
- 1: Analog watchdog interrupt enabled

Bit 5 EOCSIE: Interrupt enable for EOC

This bit is set and cleared by software to enable/disable the end of conversion interrupt.

- 0: EOC interrupt disabled
- 1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Bits 4:0 AWDCH[4:0]: Analog watchdog channel select bits

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

Note: 00000: ADC analog input Channel0
00001: ADC analog input Channel1

...

01111: ADC analog input Channel15
10000: ADC analog input Channel16

10001: ADC analog input Channel17

10010: ADC analog input Channel18

Other values reserved

10.13.3 ADC control register 2 (ADC_CR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserve d	SWST ART	EXTEN		EXTSEL[3:0]				reserve d	JSWST ART	JEXTEN		JEXTSEL[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved			ALIGN	EOCS	DDS	DMA	Reserved						CONT	ADON	
			rw	rw	rw	rw							rw	rw	

Bit 31 Reserved, must be kept at reset value.

Bit 30 SWSTART: Start conversion of regular channels

This bit is set by software to start conversion and cleared by hardware as soon as the conversion starts.

0: Reset state

1: Starts conversion of regular channels

Note: This bit can be set only when ADON = 1 otherwise no conversion is launched.

Bits 29:28 EXTEN: External trigger enable 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: Trigger detection disabled

01: Trigger detection on the rising edge

10: Trigger detection on the falling edge

11: Trigger detection on both the rising and falling edges

Bits 27:24 **EXTSEL[3:0]:** External event select for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

- 0000: Timer 1 CC1 event
- 0001: Timer 1 CC2 event
- 0010: Timer 1 CC3 event
- 0011: Timer 2 CC2 event
- 0100: Timer 2 CC3 event
- 0101: Timer 2 CC4 event
- 0110: Timer 2 TRGO event
- 0111: Timer 3 CC1 event
- 1000: Timer 3 TRGO event
- 1001: Timer 4 CC4 event
- 1010: Timer 5 CC1 event
- 1011: Timer 5 CC2 event
- 1100: Timer 5 CC3 event
- 1101: Timer 8 CC1 event
- 1110: Timer 8 TRGO event
- 1111: EXTI line11

Bit 23 Reserved, must be kept at reset value.

Bit 22 **JSWSTART:** Start conversion of injected channels

This bit is set by software and cleared by hardware as soon as the conversion starts.

0: Reset state

1: Starts conversion of injected channels

Note: This bit can be set only when ADON = 1 otherwise no conversion is launched.

Bits 21:20 **JEXTEN:** External trigger enable 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: Trigger detection disabled

01: Trigger detection on the rising edge

10: Trigger detection on the falling edge

11: Trigger detection on both the rising and falling edges

Bits 19:16 **JEXTSEL[3:0]:** External event select for injected group

These bits select the external event used to trigger the start of conversion of an injected group.

0000: Timer 1 CC4 event

0001: Timer 1 TRGO event

0010: Timer 2 CC1 event

0011: Timer 2 TRGO event

0100: Timer 3 CC2 event

0101: Timer 3 CC4 event

0110: Timer 4 CC1 event

0111: Timer 4 CC2 event

1000: Timer 4 CC3 event

1001: Timer 4 TRGO event

1010: Timer 5 CC4 event

1011: Timer 5 TRGO event

1100: Timer 8 CC2 event

1101: Timer 8 CC3 event

1110: Timer 8 CC4 event

1111: EXTI line15

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

Bit 11 **ALIGN:** Data alignment

This bit is set and cleared by software. Refer to [Figure 32](#) and [Figure 33](#).

- 0: Right alignment
- 1: Left alignment

Bit 10 **EOCS:** End of conversion selection

This bit is set and cleared by software.

- 0: The EOC bit is set at the end of each sequence of regular conversions. Overrun detection is enabled only if DMA=1.
- 1: The EOC bit is set at the end of each regular conversion. Overrun detection is enabled.

Bit 9 **DDS:** DMA disable selection (for single ADC mode)

This bit is set and cleared by software.

- 0: No new DMA request is issued after the last transfer (as configured in the DMA controller)
- 1: DMA requests are issued as long as data are converted and DMA=1

Bit 8 **DMA:** Direct memory access mode (for single ADC mode)

This bit is set and cleared by software. Refer to the DMA controller chapter for more details.

- 0: DMA mode disabled
- 1: DMA mode enabled

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **CONT:** Continuous conversion

This bit is set and cleared by software. If it is set, conversion takes place continuously until it is cleared.

- 0: Single conversion mode
- 1: Continuous conversion mode

Bit 0 **ADON:** A/D Converter ON / OFF

This bit is set and cleared by software.

- Note:*
- 0: Disable ADC conversion and go to power down mode
 - 1: Enable ADC

10.13.4 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]		
				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.

Note:

- 000: 3 cycles
- 001: 15 cycles
- 010: 28 cycles
- 011: 56 cycles
- 100: 84 cycles
- 101: 112 cycles
- 110: 144 cycles
- 111: 480 cycles

10.13.5 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:0 **SMPx[2:0]:** Channel x 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.

Note:

- 000: 3 cycles
- 001: 15 cycles
- 010: 28 cycles
- 011: 56 cycles
- 100: 84 cycles
- 101: 112 cycles
- 110: 144 cycles
- 111: 480 cycles

10.13.6 ADC injected channel data offset register x (ADC_JOFR_x)(x=1..4)

Address offset: 0x14-0x20

Reset value: 0x0000 0000

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

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

Bits 11:0 **JOFFSET_x[11:0]**: Data offset for injected channel x

These bits are written by software to define the offset to be subtracted from the raw converted data when converting injected channels. The conversion result can be read from in the ADC_JDR_x registers.

10.13.7 ADC watchdog higher threshold register (ADC_HTR)

Address offset: 0x24

Reset value: 0x0000 0FFF

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

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

Bits 11:0 **HT[11:0]**: Analog watchdog higher threshold

These bits are written by software to define the higher threshold for the analog watchdog.

10.13.8 ADC watchdog lower threshold register (ADC_LTR)

Address offset: 0x28

Reset value: 0x0000 0000

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

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

Bits 11:0 **LT[11:0]**: Analog watchdog lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

10.13.9 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16_0	SQ15[4:0]						SQ14[4:0]				SQ13[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **L[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

0000: 1 conversion
0001: 2 conversions
...

1111: 16 conversions

Bits 19:15 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0..18) assigned as the 16th in the conversion sequence.

Bits 14:10 **SQ15[4:0]:** 15th conversion in regular sequence

Bits 9:5 **SQ14[4:0]:** 14th conversion in regular sequence

Bits 4:0 **SQ13[4:0]:** 13th conversion in regular sequence

10.13.10 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10_0	SQ9[4:0]						SQ8[4:0]				SQ7[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:26 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (0..18) assigned as the 12th in the sequence to be converted.

Bits 24:20 **SQ11[4:0]:** 11th conversion in regular sequence

Bits 19:15 **SQ10[4:0]:** 10th conversion in regular sequence

Bits 14:10 **SQ9[4:0]:** 9th conversion in regular sequence

Bits 9:5 **SQ8[4:0]**: 8th conversion in regular sequence

Bits 4:0 **SQ7[4:0]**: 7th conversion in regular sequence

10.13.11 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x34

Reset value: 0x0000 0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	SQ6[4:0]					SQ5[4:0]					SQ4[4:1]					
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4_0	SQ3[4:0]					SQ2[4:0]					SQ1[4:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:25 **SQ6[4:0]**: 6th conversion in regular sequence

These bits are written by software with the channel number (0..18) assigned as the 6th in the sequence to be converted.

Bits 24:20 **SQ5[4:0]**: 5th conversion in regular sequence

Bits 19:15 **SQ4[4:0]**: 4th conversion in regular sequence

Bits 14:10 **SQ3[4:0]**: 3rd conversion in regular sequence

Bits 9:5 **SQ2[4:0]**: 2nd conversion in regular sequence

Bits 4:0 **SQ1[4:0]**: 1st conversion in regular sequence

10.13.12 ADC injected sequence register (ADC_JSQR)

Address offset: 0x38

Reset value: 0x0000 0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved											JL[1:0]		JSQ4[4:1]				
	rw	rw	rw	rw	rw	rw	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	
JSQ4[0]	JSQ3[4:0]					JSQ2[4:0]					JSQ1[4:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **JL[1:0]**: Injected sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

- Bits 19:15 **JSQ4[4:0]**: 4th conversion in injected sequence (when JL[1:0]=3, see note below)
 These bits are written by software with the channel number (0..18) assigned as the 4th in the sequence to be converted.
- Bits 14:10 **JSQ3[4:0]**: 3rd conversion in injected sequence (when JL[1:0]=3, see note below)
- Bits 9:5 **JSQ2[4:0]**: 2nd conversion in injected sequence (when JL[1:0]=3, see note below)
- Bits 4:0 **JSQ1[4:0]**: 1st conversion in injected sequence (when JL[1:0]=3, see note below)

Note: When JL[1:0]=3 (4 injected conversions in the sequencer), the ADC converts the channels in the following order: JSQ1[4:0], JSQ2[4:0], JSQ3[4:0], and JSQ4[4:0].

When JL=2 (3 injected conversions in the sequencer), the ADC converts the channels in the following order: JSQ2[4:0], JSQ3[4:0], and JSQ4[4:0].

When JL=1 (2 injected conversions in the sequencer), the ADC converts the channels in starting from JSQ3[4:0], and then JSQ4[4:0].

When JL=0 (1 injected conversion in the sequencer), the ADC converts only JSQ4[4:0] channel.

10.13.13 ADC injected data register x (ADC_JDRx) (x= 1..4)

Address offset: 0x3C - 0x48

Reset value: 0x0000 0000

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

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

Bits 15:0 **JDATA[15:0]**: Injected data

These bits are read-only. They contain the conversion result from injected channel x. The data are left -or right-aligned as shown in [Figure 32](#) and [Figure 33](#).

10.13.14 ADC regular data register (ADC_DR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

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

Bits 15:0 **DATA[15:0]**: Regular data

These bits are read-only. They contain the conversion result from the regular channels. The data are left- or right-aligned as shown in [Figure 32](#) and [Figure 33](#).

10.13.15 ADC Common status register (ADC_CSR)

Address offset: 0x00 (this offset address is relative to ADC1 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 it to 0 in the corresponding ADC_SR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved										OVR3	STRT3	JSTRT3	JEOC 3	EOC3	AWD3	
ADC3										r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	OVR2	STRT2	JSTRT2	JEOC2	EOC2	AWD2	Reserved	OVR1	STRT1	JSTRT1	JEOC 1	EOC1	AWD1			
	ADC2							ADC1						r	r	
	r	r	r	r	r	r		r	r	r	r	r	r	r	r	

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OVR3:** Overrun flag of ADC3

This bit is a copy of the OVR bit in the ADC3_SR register.

Bit 20 **STRT3:** Regular channel Start flag of ADC3

This bit is a copy of the STRT bit in the ADC3_SR register.

Bit 19 **JSTRT3:** Injected channel Start flag of ADC3

This bit is a copy of the JSTRT bit in the ADC3_SR register.

Bit 18 **JEOC3:** Injected channel end of conversion of ADC3

This bit is a copy of the JEOC bit in the ADC3_SR register.

Bit 17 **EOC3:** End of conversion of ADC3

This bit is a copy of the EOC bit in the ADC3_SR register.

Bit 16 **AWD3:** Analog watchdog flag of ADC3

This bit is a copy of the AWD bit in the ADC3_SR register.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OVR2:** Overrun flag of ADC2

This bit is a copy of the OVR bit in the ADC2_SR register.

Bit 12 **STRT2:** Regular channel Start flag of ADC2

This bit is a copy of the STRT bit in the ADC2_SR register.

Bit 11 **JSTRT2:** Injected channel Start flag of ADC2

This bit is a copy of the JSTRT bit in the ADC2_SR register.

Bit 10 **JEOC2:** Injected channel end of conversion of ADC2

This bit is a copy of the JEOC bit in the ADC2_SR register.

Bit 9 **EOC2:** End of conversion of ADC2

This bit is a copy of the EOC bit in the ADC2_SR register.

Bit 8 **AWD2:** Analog watchdog flag of ADC2

This bit is a copy of the AWD bit in the ADC2_SR register.

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

Bit 5 **OVR1:** Overrun flag of ADC1

This bit is a copy of the OVR bit in the ADC1_SR register.

Bit 4 **STRT1:** Regular channel Start flag of ADC1

This bit is a copy of the STRT bit in the ADC1_SR register.

Bit 3 **JSTRT1:** Injected channel Start flag of ADC1

This bit is a copy of the JSTRT bit in the ADC1_SR register.

Bit 2 **JEOC1:** Injected channel end of conversion of ADC1

This bit is a copy of the JEOC bit in the ADC1_SR register.

Bit 1 **EOC1:** End of conversion of ADC1

This bit is a copy of the EOC bit in the ADC1_SR register.

Bit 0 **AWD1:** Analog watchdog flag of ADC1

This bit is a copy of the AWD bit in the ADC1_SR register.

10.13.16 ADC common control register (ADC_CCR)

Address offset: 0x04 (this offset address is relative to ADC1 base address + 0x300)

Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TSVREFE:** Temperature sensor and V_{REFINT} enable

This bit is set and cleared by software to enable/disable the temperature sensor and the V_{REFINT} channel.

0: Temperature sensor and V_{REFINT} channel disabled

1: Temperature sensor and V_{REFINT} channel enabled

Bit 22 **VBATE:** 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

Bits 21:18 Reserved, must be kept at reset value.

Bits 17:16 **ADCPRE:** ADC prescaler

Set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

Note: 00: PCLK2 divided by 2

01: PCLK2 divided by 4

10: PCLK2 divided by 6

11: PCLK2 divided by 8

Bits 15:14 **DMA:** Direct memory access mode for multi ADC mode

This bit-field is set and cleared by software. Refer to the DMA controller section for more details.

00: DMA mode disabled

01: DMA mode 1 enabled (2 / 3 half-words one by one - 1 then 2 then 3)

10: DMA mode 2 enabled (2 / 3 half-words by pairs - 2&1 then 1&3 then 3&2)

11: DMA mode 3 enabled (2 / 3 bytes by pairs - 2&1 then 1&3 then 3&2)

Bit 13 **DDS:** DMA disable selection (for multi-ADC mode)

This bit is set and cleared by software.

0: No new DMA request is issued after the last transfer (as configured in the DMA controller). DMA bits are not cleared by hardware, however they must have been cleared and set to the wanted mode by software before new DMA requests can be generated.

1: DMA requests are issued as long as data are converted and DMA = 01, 10 or 11.

Bit 12 Reserved, must be kept at reset value.

Bit 11:8 **DELAY:** Delay between 2 sampling phases

Set and cleared by software. These bits are used in dual or triple interleaved modes.

0000: 5 * T_{ADCCLK}

0001: 6 * T_{ADCCLK}

0010: 7 * T_{ADCCLK}

...

1111: 20 * T_{ADCCLK}

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

Bits 4:0 **MULTI[4:0]:** Multi 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, ADC1 and ADC2 working together, ADC3 is independent

00001: Combined regular simultaneous + injected simultaneous mode

00010: Combined regular simultaneous + alternate trigger mode

00011: Reserved

00101: Injected simultaneous mode only

00110: Regular simultaneous mode only

00111: interleaved mode only

01001: Alternate trigger mode only

– 10001 to 11001: Triple mode: ADC1, 2 and 3 working together

10001: Combined regular simultaneous + injected simultaneous mode

10010: Combined regular simultaneous + alternate trigger mode

10011: Reserved

10101: Injected simultaneous mode only

10110: Regular simultaneous mode only

10111: interleaved mode only

11001: Alternate trigger mode only

All other combinations are reserved and must not be programmed

Note: In multi mode, a change of channel configuration generates an abort that can cause a loss of synchronization. It is recommended to disable the multi ADC mode before any configuration change.

10.13.17 ADC common regular data register for dual and triple modes (ADC_CDR)

Address offset: 0x08 (this offset address is relative to ADC1 base address + 0x300)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA2[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
DATA1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **DATA2[15:0]**: 2nd data item of a pair of regular conversions

- In dual mode, these bits contain the regular data of ADC2. Refer to [Dual ADC mode](#).
- In triple mode, these bits contain alternatively the regular data of ADC2, ADC1 and ADC3. Refer to [Triple ADC mode](#).

Bits 15:0 **DATA1[15:0]**: 1st data item of a pair of regular conversions

- In dual mode, these bits contain the regular data of ADC1. Refer to [Dual ADC mode](#)
- In triple mode, these bits contain alternatively the regular data of ADC1, ADC3 and ADC2. Refer to [Triple ADC mode](#).

10.13.18 ADC register map

The following table summarizes the ADC registers.

Table 38. ADC global register map

Offset	Register
0x000 - 0x04C	ADC1
0x050 - 0x0FC	Reserved
0x100 - 0x14C	ADC2
0x118 - 0x1FC	Reserved
0x200 - 0x24C	ADC3
0x250 - 0x2FC	Reserved
0x300 - 0x308	Common registers

Table 39. ADC register map and reset values for each ADC

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_SR	Reserved																				OVR	STRT	JSTRT	JEOC	EOC	AWD	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				
0x04	ADC_CR1	Reserved				OVRIE	RES[1:0]	AWDEN		JAWDEN		Reserved				DISC NUM [2:0]	JDISCEN				AWDCH[4:0]				0	0	0	0	0	0			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x08	ADC_CR2	Re se rv ed	SWSTART	EXTEN[1:0]	EXTSEL [3:0]			Re se rv ed	JSWSTART	JEXTEN[1:0]	JEXTSEL [3:0]			Reserved				ALIGN	DISCEN	JAUTO	AWD_SGL	SCAN	JEOCIE	AWDIE	CONT				ADON	0	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				
0x0C	ADC_SMPR1	Sample time bits SMPx_X																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	ADC_SMPR2	Sample time bits SMPx_X																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x14	ADC_JOFR1	Reserved												JOFFSET1[11:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x18	ADC_JOFR2	Reserved												JOFFSET2[11:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	ADC_JOFR3	Reserved												JOFFSET3[11:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x20	ADC_JOFR4	Reserved												JOFFSET4[11:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x24	ADC_HTR	Reserved												HT[11:0]																			
	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				
0x28	ADC_LTR	Reserved												LT[11:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x2C	ADC_SQR1	Reserved				L[3:0]			Regular channel sequence SQx_X bits																								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x30	ADC_SQR2	Reserved	Regular channel sequence SQx_X bits																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x34	ADC_SQR3	Reserved	Regular channel sequence SQx_X bits																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x38	ADC_JSQR	Reserved				JL[1:0]		Injected channel sequence JSQx_X bits																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x3C	ADC_JDR1	Reserved												JDATA[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				
0x40	ADC_JDR2	Reserved												JDATA[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				
0x44	ADC_JDR3	Reserved												JDATA[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				
0x48	ADC_JDR4	Reserved												JDATA[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				
0x4C	ADC_DR	Reserved												Regular DATA[15:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Table 40. ADC register map and reset values (common ADC registers)

Refer to [Table 1 on page 50](#) for the register boundary addresses.

11 Digital-to-analog converter (DAC)

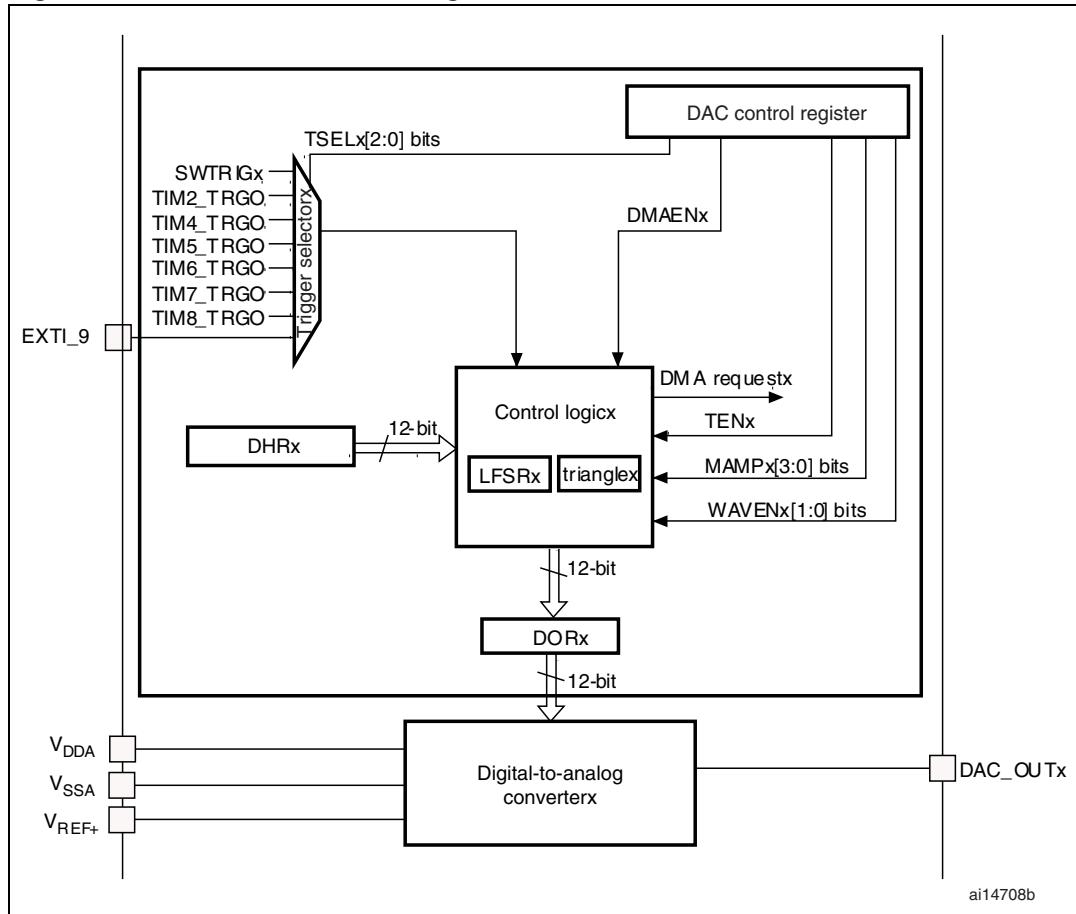
11.1 DAC introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC has two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, V_{REF+} (shared with ADC) is available for better resolution.

11.2 DAC main features

- Two DAC converters: one output channel each
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave generation
- Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel
- DMA underrun error detection
- External triggers for conversion
- Input voltage reference, V_{REF+}

Figure 48 shows the block diagram of a DAC channel and *Table 41* gives the pin description.

Figure 48. DAC channel block diagram**Table 41.** DAC pins

Name	Signal type	Remarks
V _{REF+}	Input, analog reference positive	The higher/positive reference voltage for the DAC, $1.8 \text{ V} \leq V_{\text{REF}+} \leq V_{\text{DDA}}$
V _{DDA}	Input, analog supply	Analog power supply
V _{SSA}	Input, analog supply ground	Ground for analog power supply
DAC_OUTx	Analog output signal	DAC channelx analog output

Note: Once the DAC channelx is enabled, the corresponding GPIO pin (PA4 or PA5) is automatically connected to the analog converter output (DAC_OUTx). In order to avoid parasitic consumption, the PA4 or PA5 pin should first be configured to analog (AIN).

11.3 DAC functional description

11.3.1 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a startup time t_{WAKEUP}

Note: The ENx bit enables the analog DAC Channelx macrocell only. The DAC Channelx digital interface is enabled even if the ENx bit is reset.

11.3.2 DAC output buffer enable

The DAC integrates two output buffers that can be used to reduce the output impedance, and to drive external loads directly without having to add an external operational amplifier. Each DAC channel output buffer can be enabled and disabled using the corresponding BOFFx bit in the DAC_CR register.

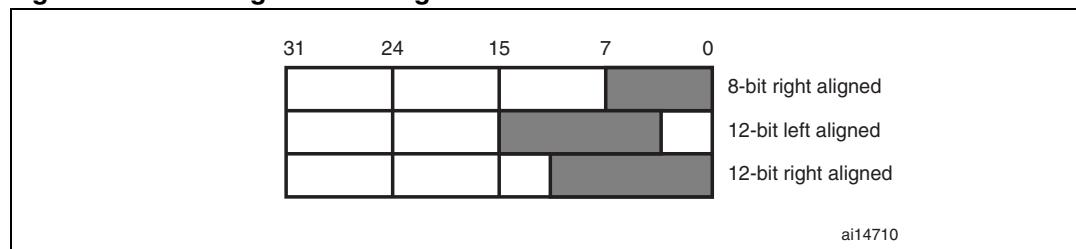
11.3.3 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channelx, there are three possibilities:
 - 8-bit right alignment: the software has to load data into the DAC_DHR8Rx [7:0] bits (stored into the DHRx[11:4] bits)
 - 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)
 - 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyx register, the data written by the user is shifted and stored into the corresponding DHRx (data holding registerx, which are internal non-memory-mapped registers). The DHRx register is then loaded into the DORx register either automatically, by software trigger or by an external event trigger.

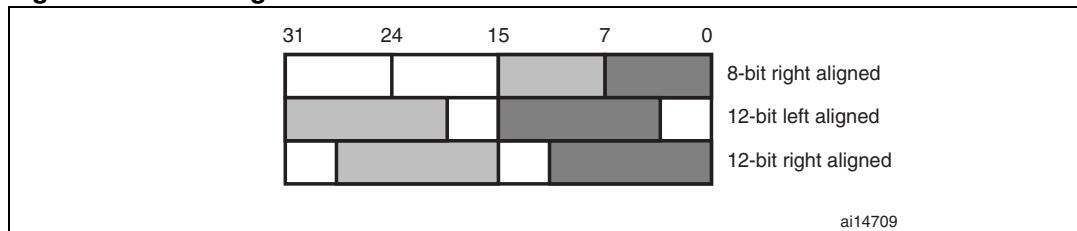
Figure 49. Data registers in single DAC channel mode



- Dual DAC channels, there are three possibilities:
 - 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)
 - 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)
 - 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 50. Data registers in dual DAC channel mode



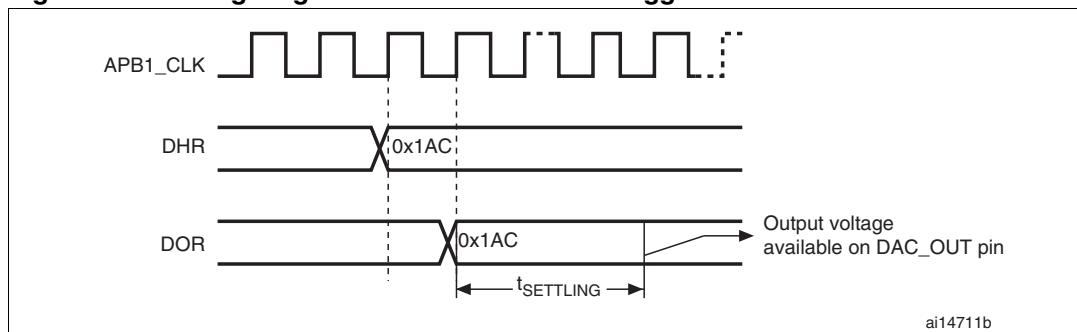
11.3.4 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12LD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one APB1 clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three APB1 clock cycles later.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

Figure 51. Timing diagram for conversion with trigger disabled TEN = 0



11.3.5 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+} .

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DACOutput} = V_{REF} \times \frac{\text{DOR}}{4095}$$

11.3.6 DAC trigger selection

If the TENx control bit is set, conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[2:0] control bits determine which out of 8 possible events will trigger conversion as shown in [Table 42](#).

Table 42. External triggers

Source	Type	TSEL[2:0]
Timer 6 TRGO event	Internal signal from on-chip timers	000
Timer 8 TRGO event		001
Timer 7 TRGO event		010
Timer 5 TRGO event		011
Timer 2 TRGO event		100
Timer 4 TRGO event		101
EXTI line9	External pin	110
SWTRIG	Software control bit	111

Each time a DAC interface detects a rising edge on the selected timer TRGO output, or on the selected external interrupt line 9, the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three APB1 cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

- Note:*
- 1 *TSELx[2:0] bit cannot be changed when the ENx bit is set.*
 - 2 *When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one APB1 clock cycle.*

11.3.7 DMA request

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

A DAC DMA request is generated when an external trigger (but not a software trigger) occurs while the DMAENx bit is set. The value of the DAC_DHRx register is then transferred into the DAC_DORx register.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, you should set only the corresponding DMAENx bit. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgement for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. DMA data transfers are then disabled and no further DMA request is treated. The DAC channelx continues to convert old data.

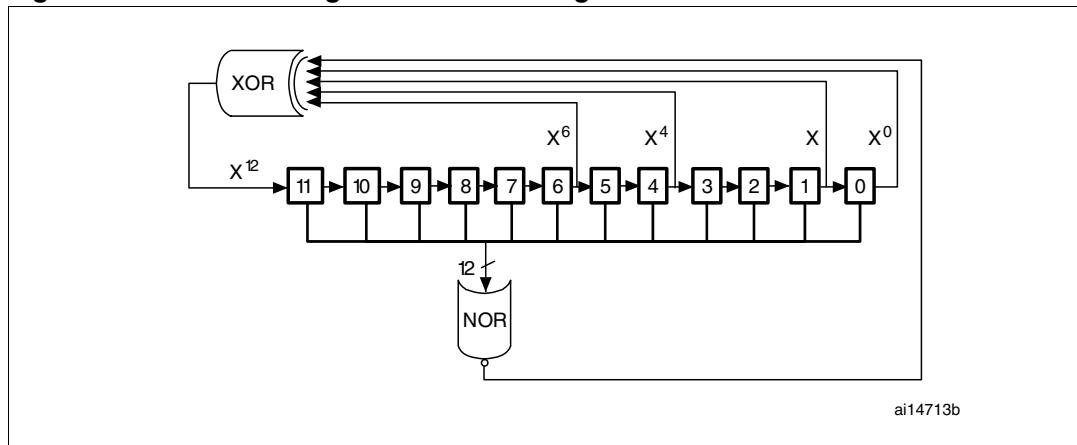
The software should clear the DMAUDRx flag by writing “1”, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software should modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channlex, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

11.3.8 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to “01”. The preloaded value in LFSR is 0xAAA. This register is updated three APB1 clock cycles after each trigger event, following a specific calculation algorithm.

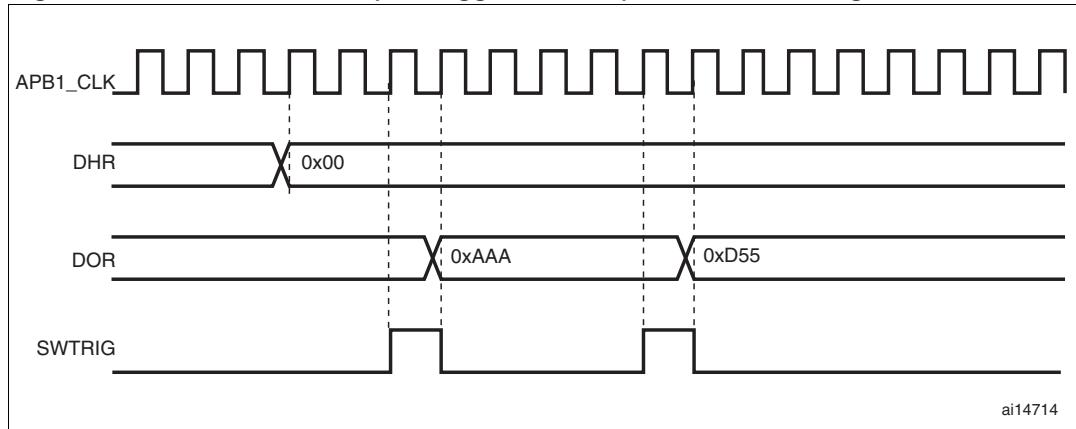
Figure 52. DAC LFSR register calculation algorithm



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 53. DAC conversion (SW trigger enabled) with LFSR wave generation

Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

11.3.9 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to “10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three APB1 clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is stored into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

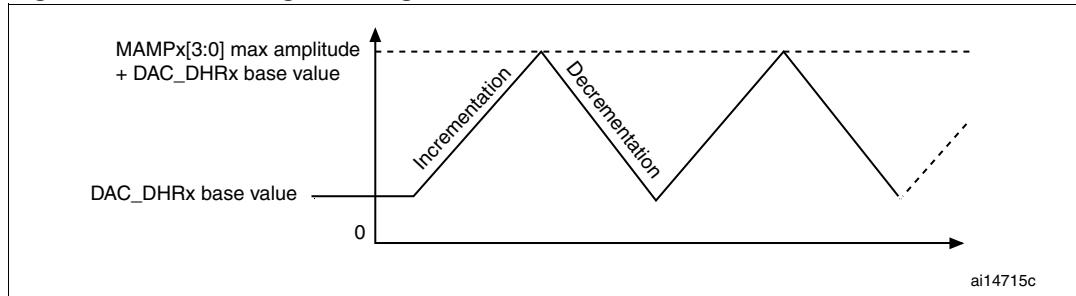
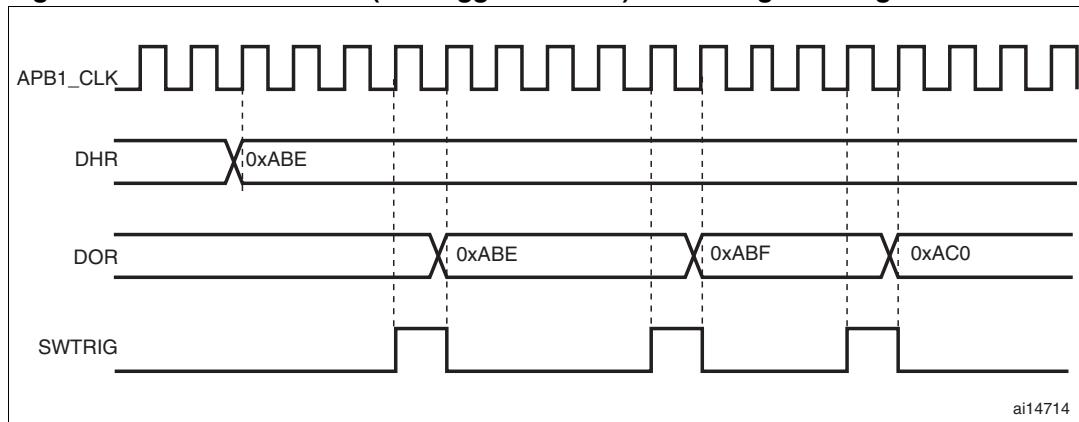
Figure 54. DAC triangle wave generation

Figure 55. DAC conversion (SW trigger enabled) with triangle wave generation

ai14714

- Note:**
- 1 *The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.*
 - 2 *The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.*

11.4 Dual DAC channel conversion

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time.

Eleven possible conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

11.4.1 Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three APB1 clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three APB1 clock cycles later).

11.4.2 Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated.

11.4.3 Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated.

11.4.4 Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and the same maximum amplitude value in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into

DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

11.4.5 Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

11.4.6 Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

In this configuration, one APB1 clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

11.4.7 Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three APB1 clock cycles).

11.4.8 Simultaneous trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The LFSR2 counter is then updated.

11.4.9 Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated.

At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The LFSR2 counter is then updated.

11.4.10 Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and the same maximum amplitude value using the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is

added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

11.4.11 Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVE[1:0] bits as “1x” and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the DAC channel2 triangle counter is updated.

11.5 DAC registers

Refer to [Section 1.1 on page 46](#) for a list of abbreviations used in register descriptions.

11.5.1 DAC control register (DAC_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]				TEN2	BOFF2	EN2
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]				TEN1	BOFF1	EN1
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled

1: DAC channel2 DMA mode enabled

Bit 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

- 0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
- 0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
- 0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
- 0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
- 0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
- 0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
- 0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
- 0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
- 1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
- 1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
- 1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
- ≥ 1011 : Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bit 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.

- 00: wave generation disabled
- 01: Noise wave generation enabled
- 1x: Triangle wave generation enabled

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)

Bits 21:19 **TSEL2[2:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

- 000: Timer 6 TRGO event
- 001: Timer 8 TRGO event
- 010: Timer 7 TRGO event
- 011: Timer 5 TRGO event
- 100: Timer 2 TRGO event
- 101: Timer 4 TRGO event
- 110: External line9
- 111: Software trigger

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).

Bit 18 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

- 0: DAC channel2 trigger disabled and data written into the DAC_DHRx register are transferred one APB1 clock cycle later to the DAC_DOR2 register
- 1: DAC channel2 trigger enabled and data from the DAC_DHRx register are transferred three APB1 clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DOR2 register takes only one APB1 clock cycle.

Bit 17 **BOFF2**: DAC channel2 output buffer disable

This bit is set and cleared by software to enable/disable DAC channel2 output buffer.

- 0: DAC channel2 output buffer enabled
- 1: DAC channel2 output buffer disabled

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

- 0: DAC channel2 disabled
- 1: DAC channel2 enabled

Bits 15:14 Reserved, 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 8 TRGO event

010: Timer 7 TRGO event

011: Timer 5 TRGO event

100: Timer 2 TRGO event

101: Timer 4 TRGO event

110: External line9

111: Software trigger

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 2 **TEN1**: DAC channel1 trigger enable

This bit is set and cleared by software to enable/disable DAC channel1 trigger.

0: DAC channel1 trigger disabled and data written into the DAC_DHRx register are transferred one APB1 clock cycle later to the DAC_DOR1 register

1: DAC channel1 trigger enabled and data from the DAC_DHRx register are transferred three APB1 clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DOR1 register takes only one APB1 clock cycle.

Bit 1 **BOFF1**: DAC channel1 output buffer disable

This bit is set and cleared by software to enable/disable DAC channel1 output buffer.

0: DAC channel1 output buffer enabled

1: DAC channel1 output buffer disabled

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.

0: DAC channel1 disabled

1: DAC channel1 enabled

11.5.2 DAC software trigger register (DAC_SWTRIGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
														SWTRIG2	SWTRIG1
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set and cleared by software to enable/disable the software trigger.

0: Software trigger disabled

1: Software trigger enabled

Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set and cleared by software to enable/disable the software trigger.

0: Software trigger disabled

1: Software trigger enabled

Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

11.5.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

11.5.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Reserved

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

Bit 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

11.5.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[7:0]															
Reserved								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.

11.5.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]															
Reserved				rw											

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

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

11.5.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]															
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 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data

These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

11.5.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel2.

11.5.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DACC2DHR[11:0]													
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]		DACC1DHR[11:0]													
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

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

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

11.5.10 DUAL DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]															
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
DACC1DHR[11:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

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

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

11.5.11 DUAL DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel1.

11.5.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DOR[11:0]															
Reserved				r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 11:0 **DACC1DOR[11:0]**: DAC channel1 data output

These bits are read-only, they contain data output for DAC channel1.

11.5.13 DAC channel2 data output register (DAC_DOR2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DOR[11:0]															
Reserved				r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 11:0 **DACC2DOR[11:0]**: DAC channel2 data output

These bits are read-only, they contain data output for DAC channel2.

11.5.14 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
DMAUDR2				Reserved											
rc_w1															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
DMAUDR1				Reserved											
rc_w1				Reserved											

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

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate)

Bits 28:14 Reserved, must be kept at reset value.

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bits 12:0 Reserved, must be kept at reset value.

11.5.15 DAC register map

Table 43 summarizes the DAC registers.

Table 43. DAC register map

Address offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]	WAVE2[2:0]	TSEL2[2:0]		TEN2	BOFF2	EN2	Reserved	DMAUDRIE1	DMAEN1	MAMP1[3:0]	WAVE1[2:0]	TSEL1[2:0]	TEN1	SWTRIG2	BOFF1	EN1											
0x04	DAC_SWT_RIGR													Reserved																			
0x08	DAC_DHR1_2R1													Reserved																			
0x0C	DAC_DHR1_2L1													Reserved																			
0x10	DAC_DHR8_R1													Reserved																			
0x14	DAC_DHR1_2R2													Reserved																			
0x18	DAC_DHR1_2L2													Reserved																			
0x1C	DAC_DHR8_R2													Reserved																			
0x20	DAC_DHR1_2RD	Reserved												DACC2DHR[11:0]		Reserved																	
0x24	DAC_DHR1_2LD													DACC2DHR[11:0]		Reserved																	
0x28	DAC_DHR8_RD													Reserved																			
0x2C	DAC_DOR1													Reserved																			
0x30	DAC_DOR2													Reserved																			
0x34	DAC_SR	Reserved	DMAUDR2											Reserved		DMAUDR1																	

Refer to *Table 1 on page 50* for the register boundary addresses.

12 Digital camera interface (DCMI)

12.1 DCMI introduction

The digital camera is a synchronous parallel interface able to receive a high-speed data flow from an external 8-, 10-, 12- or 14-bit CMOS camera module. It supports different data formats: YCbCr4:2:2/RGB565 progressive video and compressed data (JPEG).

This interface is for use with black & white cameras, X24 and X5 cameras, and it is assumed that all pre-processing like resizing is performed in the camera module.

12.2 DCMI main features

- 8-, 10-, 12- or 14-bit parallel interface
- Embedded/external line and frame synchronization
- Continuous or snapshot mode
- Crop feature
- Supports the following data formats:
 - 8/10/12/14- bit progressive video: either monochrome or raw bayer
 - YCbCr 4:2:2 progressive video
 - RGB 565 progressive video
 - Compressed data: JPEG

12.3 DCMI pins

Table 44 shows the DCMI pins.

Table 44. DCMI pins

Name	Signal type
D[0:13]	Data inputs
HSYNC	Horizontal synchronization input
VSYNC	Vertical synchronization input
PIXCLX	Pixel clock input

12.4 DCMI clocks

The digital camera interface uses two clock domains PIXCLK and HCLK. The signals generated with PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from the camera are stable and can be sampled. The maximum PIXCLK period must be higher than 2.5 HCLK periods.

12.5 DCMI functional overview

The digital camera interface is a synchronous parallel interface that can receive high-speed (up to 54 Mbytes/s) data flows. It consists of up to 14 data lines (D13-D0) and a pixel clock line (PIXCLK). The pixel clock has a programmable polarity, so that data can be captured on either the rising or the falling edge of the pixel clock.

The data are packed into a 32-bit data register (DCMI_DR) and then transferred through a general-purpose DMA channel. The image buffer is managed by the DMA, not by the camera interface.

The data received from the camera can be organized in lines/frames (raw YUB/RGB/Bayer modes) or can be a sequence of JPEG images. To enable JPEG image reception, the JPEG bit (bit 3 of DCMI_CR register) must be set.

The data flow is synchronized either by hardware using the optional HSYNC (horizontal synchronization) and VSYNC (vertical synchronization) signals or by synchronization codes embedded in the data flow.

Figure 56 shows the DCMI block diagram.

Figure 56. DCMI block diagram

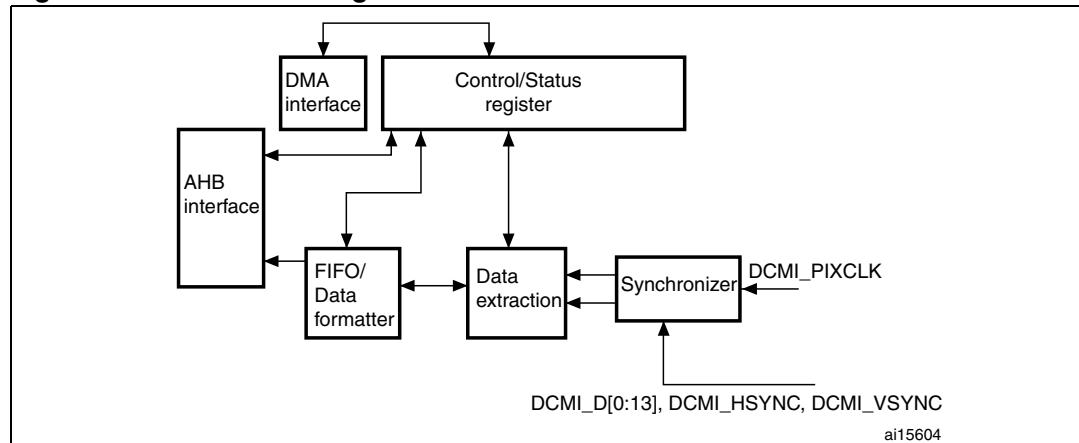
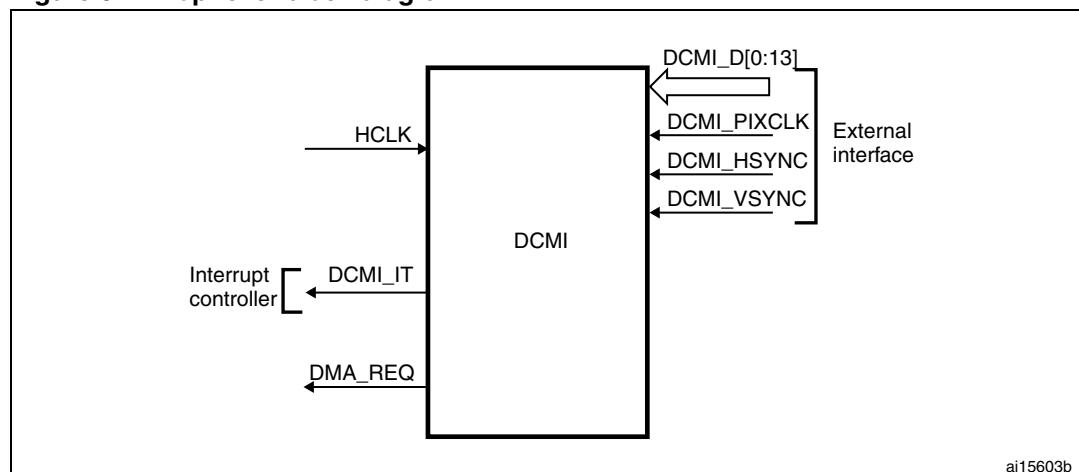


Figure 57. Top-level block diagram



12.5.1 DMA interface

The DMA interface is active when the CAPTURE bit in the DCMI_CR register is set. A DMA request is generated each time the camera interface receives a complete 32-bit data block in its register.

12.5.2 DCMI physical interface

The interface is composed of 11/13/15/17 inputs. Only the Slave mode is supported.

The camera interface can capture 8-bit, 10-bit, 12-bit or 14-bit data depending on the EDM[1:0] bits in the DCMI_CR register. If less than 14 bits are used, the unused input pins must be connected to ground.

Table 45. DCMI signals

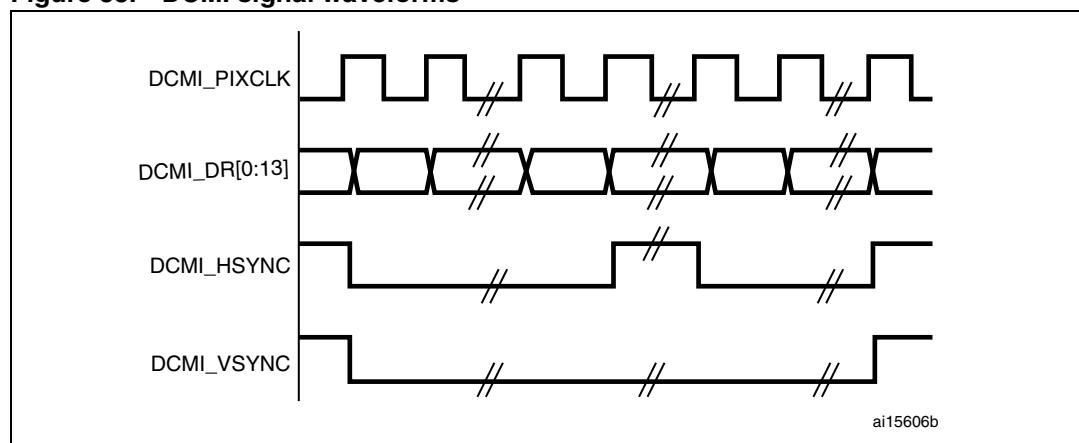
Signal name		Signal description
8 bits	D[0..7]	
10 bits	D[0..9]	
12 bits	D[0..11]	Data
14 bits	D[0..13]	
PIXCLK		Pixel clock
H SYNC		Horizontal synchronization / Data valid
V SYNC		Vertical synchronization

The data are synchronous with PIXCLK and change on the rising/falling edge of the pixel clock depending on the polarity.

The HSYNC signal indicates the start/end of a line.

The VSYNC signal indicates the start/end of a frame

Figure 58. DCMI signal waveforms



1. The capture edge of DCMI_PIXCLK is the falling edge, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
1. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

8-bit data

When EDM[1:0] in DCMI_CR are programmed to “00” the interface captures 8 LSB’s at its input (D[0:7]) and stores them as 8-bit data. The D[13:8] inputs are ignored. In this case, to capture a 32-bit word, the camera interface takes four pixel clock cycles.

The first captured data byte is placed in the LSB position in the 32-bit word and the 4th captured data byte is placed in the MSB position in the 32-bit word. [Table 46](#) gives an example of the positioning of captured data bytes in two 32-bit words.

Table 46. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D _{n+3} [7:0]	D _{n+2} [7:0]	D _{n+1} [7:0]	D _n [7:0]
4	D _{n+7} [7:0]	D _{n+6} [7:0]	D _{n+5} [7:0]	D _{n+4} [7:0]

10-bit data

When EDM[1:0] in DCMI_CR are programmed to “01”, the camera interface captures 10-bit data at its input D[0..9] and stores them as the 10 least significant bits of a 16-bit word. The remaining most significant bits in the DCMI_DR register (bits 11 to 15) are cleared to zero. So, in this case, a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in [Table 47](#).

Table 47. Positioning of captured data bytes in 32-bit words (10-bit width)

Byte address	31:26	25:16	15:10	9:0
0	0	D _{n+1} [9:0]	0	D _n [9:0]
4	0	D _{n+3} [9:0]	0	D _{n+2} [9:0]

12-bit data

When EDM[1:0] in DCMI_CR are programmed to “10”, the camera interface captures the 12-bit data at its input D[0..11] and stores them as the 12 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in [Table 48](#).

Table 48. Positioning of captured data bytes in 32-bit words (12-bit width)

Byte address	31:28	27:16	15:12	11:0
0	0	D _{n+1} [11:0]	0	D _n [11:0]
4	0	D _{n+3} [11:0]	0	D _{n+2} [11:0]

14-bit data

When EDM[1:0] in DCMI_CR are programmed to “11”, the camera interface captures the 14-bit data at its input D[0..13] and stores them as the 14 least significant bits of a 16-bit

word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in [Table 49](#).

Table 49. Positioning of captured data bytes in 32-bit words (14-bit width)

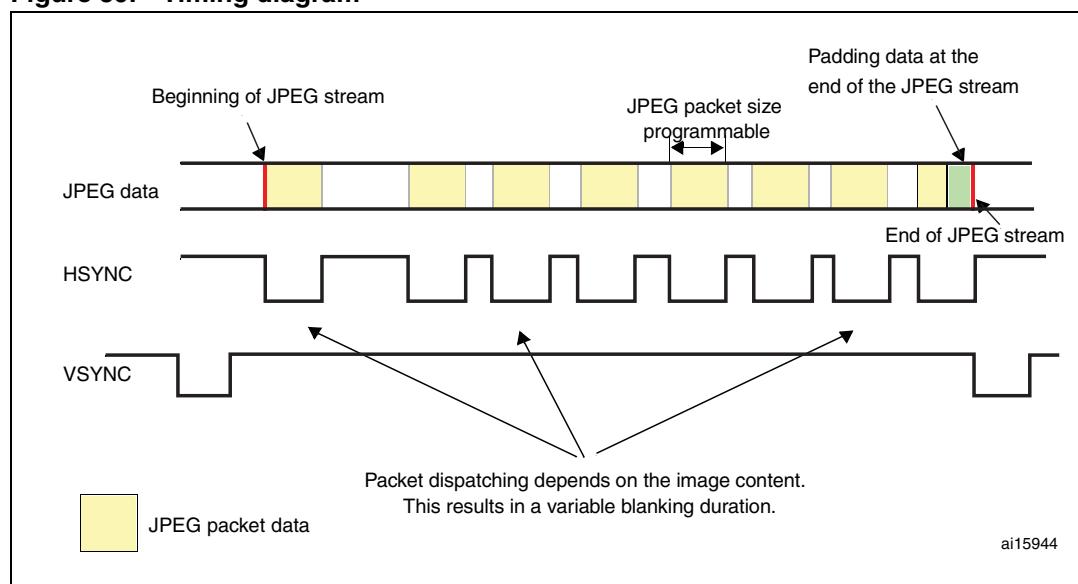
Byte address	31:30	29:16	15:14	13:0
0	0	D _{n+1} [13:0]	0	D _n [13:0]
4	0	D _{n+3} [13:0]	0	D _{n+2} [13:0]

12.5.3 Synchronization

The digital camera interface supports embedded or hardware (HSYNC & VSYNC) synchronization. When embedded synchronization is used, it is up to the digital camera module to make sure that the 0x00 and 0xFF values are used ONLY for synchronization (not in data). Embedded synchronization codes are supported only for the 8-bit parallel data interface width (that is, in the DCMI_CR register, the EDM[1:0] bits should be cleared to "00").

For compressed data, the DCMI supports only the hardware synchronization mode. In this case, VSYNC is used as a start/end of the image, and HSYNC is used as a Data Valid signal. [Figure 59](#) shows the corresponding timing diagram.

Figure 59. Timing diagram



Hardware synchronization mode

In hardware synchronisation mode, the two synchronization signals (HSYNC/VSYNC) are used.

Depending on the camera module/module, data may be transmitted during horizontal/vertical synchronisation periods. The HSYNC/VSYNC signals act like blanking signals since all the data received during HSYNC/VSYNC active periods are ignored.

In order to correctly transfer images into the DMA/RAM buffer, data transfer is synchronized with the VSYNC signal. When the hardware synchronisation mode is selected, and capture is enabled (CAPTURE bit set in DCMI_CR), data transfer is synchronized with the deactivation of the VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by DMA to successive buffers or the same/circular buffer. To allow the DMA management of successive frames, a VSIF (Vertical synchronization interrupt flag) is activated at the end of each frame.

Embedded data synchronization mode

In this synchronisation mode, the data flow is synchronised using 32-bit codes embedded in the data flow. These codes use the 0x00/0xFF values that are *not* used in data anymore.

There are 4 types of codes, all with a 0xFF0000XY format. The embedded synchronization codes are supported only in 8-bit parallel data width capture (in the DCMI_CR register, the EDM[1:0] bits should be programmed to "00"). For other data widths, this mode generates unpredictable results and must not be used.

Note:

Camera modules can have 8 such codes (in interleaved mode). For this reason, the interleaved mode is not supported by the camera interface (otherwise, every other half-frame would be discarded).

- Mode 2

Four embedded codes signal the following events

- Frame start (FS)
- Frame end (FE)
- Line start (LS)
- Line end (LE)

The XY values in the 0xFF0000XY format of the four codes are programmable (see [Section 12.8.7: DCMI embedded synchronization code register \(DCMI_ESCR\)](#)).

A 0xFF value programmed as a “frame end” means that all the unused codes are interpreted as valid frame end codes.

In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.

- Mode 1

An alternative coding is the camera mode 1. This mode is ITU656 compatible.

The codes signal another set of events:

- SAV (active line) - line start
- EAV (active line) - line end
- SAV (blanking) - end of line during interframe blanking period
- EAV (blanking) - end of line during interframe blanking period

This mode can be supported by programming the following codes:

- FS \leq 0xFF
- FE \leq 0xFF
- LS \leq SAV (active)
- LE \leq EAV (active)

An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. You can therefore select a bit to compare in the embedded code and detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

Example

FS = 0xA5

Unmask code for FS = 0x10

In this case the frame start code is embedded in the bit 4 of the frame start code.

12.5.4 Capture modes

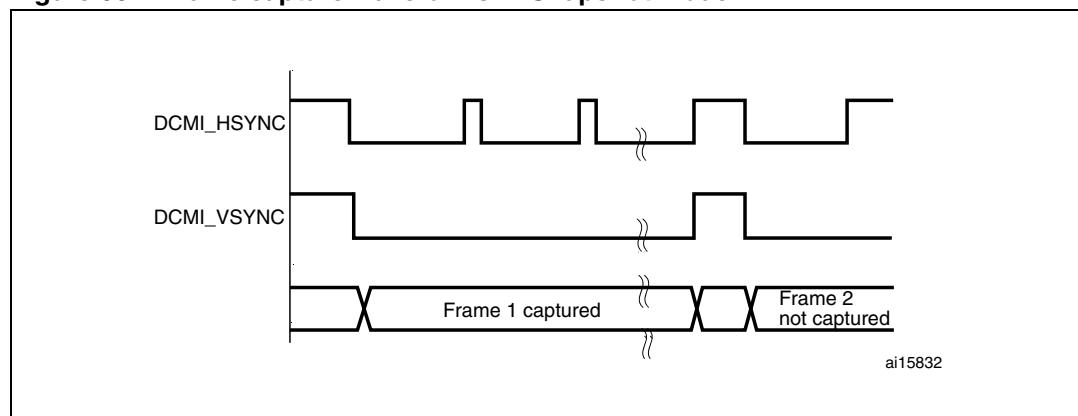
This interface supports two types of capture: snapshot (single frame) and continuous grab.

Snapshot mode (single frame)

In this mode, a single frame is captured (CM = '1' in the DCMI_CR register). After the CAPTURE bit is set in DCMI_CR, the interface waits for the detection of a start of frame before sampling the data. The camera interface is automatically disabled (CAPTURE bit cleared in DCMI_CR) after receiving the first complete frame. An interrupt is generated (IT_FRAME) if it is enabled.

In case of an overrun, the frame is lost and the CAPTURE bit is cleared.

Figure 60. Frame capture waveforms in Snapshot mode

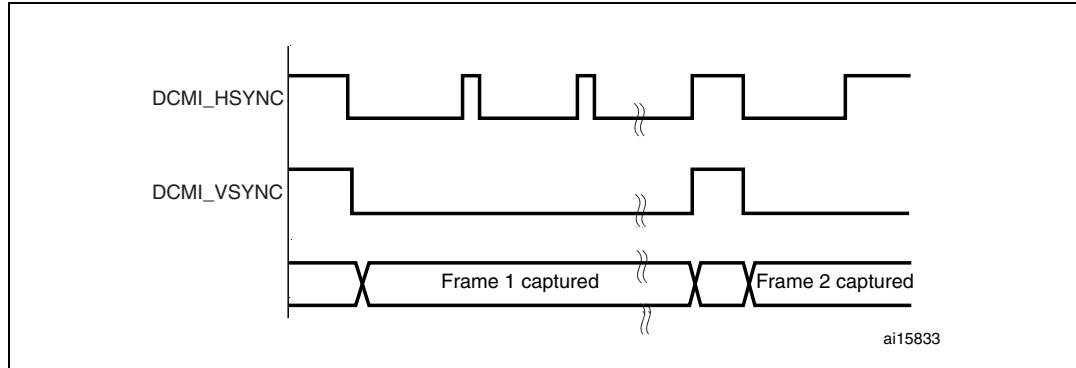


1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

Continuous grab mode

In this mode (CM bit = '0' in DCMI_CR), once the CAPTURE bit has been set in DCMI_CR, the grabbing process starts on the next VSYNC or embedded frame start depending on the mode. The process continues until the CAPTURE bit is cleared in DCMI_CR. Once the CAPTURE bit has been cleared, the grabbing process continues until the end of the current frame.

Figure 61. Frame capture waveforms in continuous grab mode



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

In continuous grab mode, you can configure the FCRC bits in DCMI_CR to grab all pictures, every second picture or one out of four pictures to decrease the frame capture rate.

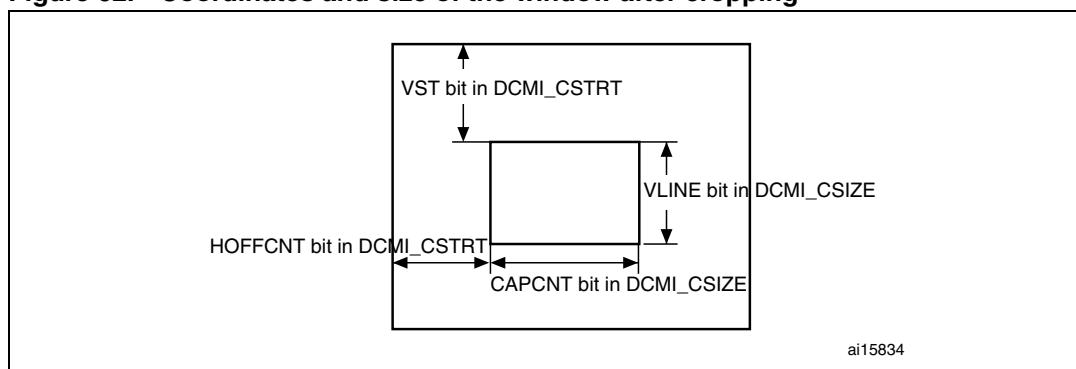
Note:

In the hardware synchronization mode (ESS = '0' in DCMI_CR), the IT_VSYNC interrupt is generated (if enabled) even when CAPTURE = '0' in DCMI_CR so, to reduce the frame capture rate even further, the IT_VSYNC interrupt can be used to count the number of frames between 2 captures in conjunction with the Snapshot mode. This is not allowed by embedded data synchronization mode.

12.5.5 Crop feature

With the crop feature, the camera interface can select a rectangular window from the received image. The start (upper left corner) coordinates and size (horizontal dimension in number of pixel clocks and vertical dimension in number of lines) are specified using two 32-bit registers (DCMI_CWSTRT and DCMI_CWSIZE). The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension).

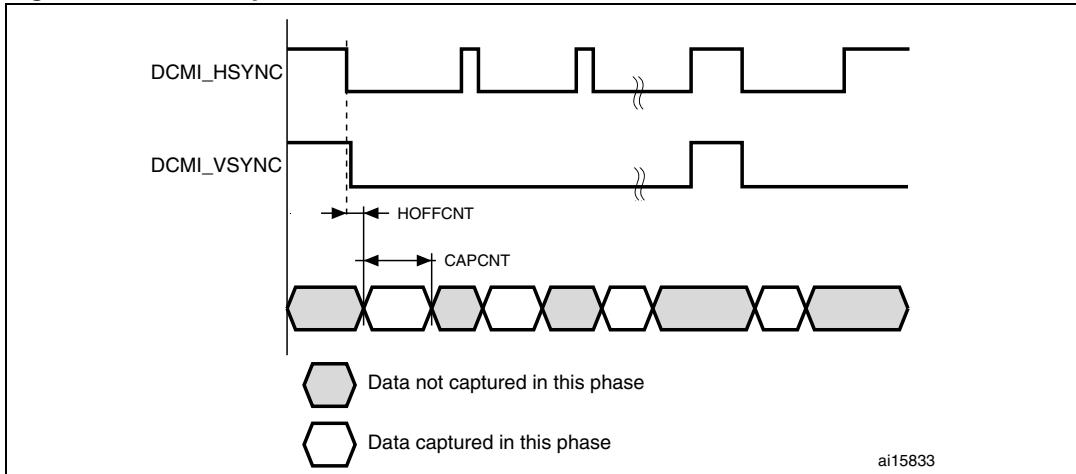
Figure 62. Coordinates and size of the window after cropping



These registers specify the coordinates of the starting point of the capture window as a line number (in the frame, starting from 0) and a number of pixel clocks (on the line, starting from 0), and the size of the window as a line number and a number of pixel clocks. The CAPCNT value can only be a multiple of 4 (two least significant bits are forced to 0) to allow the correct transfer of data through the DMA.

If the VSYNC signal goes active before the number of lines is specified in the DCMI_CWSIZE register, then the capture stops and an IT_FRAME interrupt is generated when enabled.

Figure 63. Data capture waveforms



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

ai15833

12.5.6 JPEG format

To allow JPEG image reception, it is necessary to set the JPEG bit in the DCMI_CR register. JPEG images are not stored as lines and frames, so the VSYNC signal is used to start the capture while HSYNC serves as a data enable signal. The number of bytes in a line may not be a multiple of 4, you should therefore be careful when handling this case since a DMA request is generated each time a complete 32-bit word has been constructed from the captured data. When an end of frame is detected and the 32-bit word to be transferred has not been completely received, the remaining data are padded with '0's and a DMA request is generated.

The crop feature and embedded synchronization codes cannot be used in the JPEG format.

12.5.7 FIFO

A four-word FIFO is implemented to manage data rate transfers on the AHB. The DCMI features a simple FIFO controller with a read pointer incremented each time the camera interface reads from the AHB, and a write pointer incremented each time the camera interface writes to the FIFO. There is no overrun protection to prevent the data from being overwritten if the AHB interface does not sustain the data transfer rate.

In case of overrun or errors in the synchronization signals, the FIFO is reset and the DCMI interface waits for a new start of frame.

12.6 Data format description

12.6.1 Data formats

Three types of data are supported:

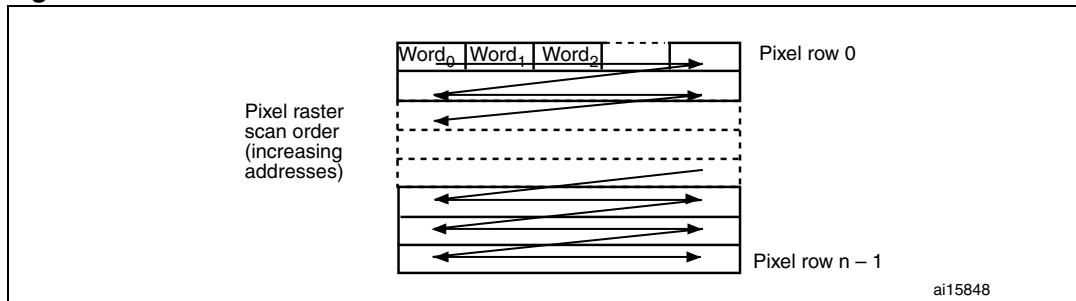
- 8-bit progressive video: either monochrome or raw Bayer format
- YCbCr 4:2:2 progressive video
- RGB565 progressive video. A pixel coded in 16 bits (5 bits for blue, 5 bits for red, 6 bits for green) takes two clock cycles to be transferred.

Compressed data: JPEG

For B&W, YCbCr or RGB data, the maximum input size is 2048×2048 pixels. No limit in JPEG compressed mode.

For monochrome, RGB & YCbCr, the frame buffer is stored in raster mode. 32-bit words are used. Only the little endian format is supported.

Figure 64. Pixel raster scan order



12.6.2 Monochrome format

Characteristics:

- Raster format
- 8 bits per pixel

[Table 50](#) shows how the data are stored.

Table 50. Data storage in monochrome progressive video format

Byte address	31:24	23:16	15:8	7:0
0	n + 3	n + 2	n + 1	n
4	n + 7	n + 6	n + 5	n + 4

12.6.3 RGB format

Characteristics:

- Raster format
- RGB
- Interleaved: one buffer: R, G & B interleaved: BRGBRBGRG, etc.
- Optimized for display output

The RGB planar format is compatible with standard OS frame buffer display formats. Only 16 BPP (bits per pixel): RGB565 (2 pixels per 32-bit word) is supported. The 24 BPP (palletized format) and grayscale formats are not supported. Pixels are stored in a raster scan order, that is from top to bottom for pixel rows, and from left to right within a pixel row. Pixel components are R (red), G (green) and B (blue). All components have the same spatial resolution (4:4:4 format). A frame is stored in a single part, with the components interleaved on a pixel basis.

Table 51 shows how the data are stored.

Table 51. Data storage in RGB progressive video format

Byte address	31:27	26:21	20:16	15:11	10:5	4:0
0	Red n + 1	Green n + 1	Blue n + 1	Red n	Green n	Blue n
4	Red n + 4	Green n + 3	Blue n + 3	Red n + 2	Green n + 2	Blue n + 2

12.6.4 YCbCr format

Characteristics:

- Raster format
- YCbCr 4:2:2
- Interleaved: one Buffer: Y, Cb & Cr interleaved: CbYCrYCbYCr, etc.

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). Each component is encoded in 8 bits. Luma and chroma are stored together (interleaved) as shown in *Table 52*.

Table 52. Data storage in YCbCr progressive video format

Byte address	31:24	23:16	15:8	7:0
0	Y n + 1	Cr n	Y n	Cb n
4	Y n + 3	Cr n + 2	Y n + 2	Cb n + 2

12.7 DCMI interrupts

Five interrupts are generated. All interrupts are maskable by software. The global interrupt (IT_DCMI) is the OR of all the individual interrupts. *Table 53* gives the list of all interrupts.

Table 53. DCMI interrupts

Interrupt name	Interrupt event
IT_LINE	Indicates the end of line
IT_FRAME	Indicates the end of frame capture
IT_OVR	indicates the overrun of data reception
IT_VSYNC	Indicates the synchronization frame
IT_ERR	Indicates the detection of an error in the embedded synchronization frame detection
IT_DCMI	Logic OR of the previous interrupts

12.8 DCMI register description

All DCMI registers have to be accessed as 32-bit words, otherwise a bus error occurs.

12.8.1 DCMI control register 1 (DCMI_CR)

Address offset: 0x00

Reset value: 0x0000 0x0000

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

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

Bit 14 **ENABLE:** DCMI enable

0: DCMI disabled

1: DCMI enabled

Note: The DCMI configuration registers should be programmed correctly before enabling this Bit

Bit 13: 12 Reserved, must be kept at reset value.

11:10 **EDM[1:0]:** Extended data mode

00: Interface captures 8-bit data on every pixel clock

01: Interface captures 10-bit data on every pixel clock

10: Interface captures 12-bit data on every pixel clock

11: Interface captures 14-bit data on every pixel clock

9:8 **FCRC[1:0]:** Frame capture rate control

These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode. They are ignored in snapshot mode.

00: All frames are captured

01: Every alternate frame captured (50% bandwidth reduction)

10: One frame in 4 frames captured (75% bandwidth reduction)

11: reserved

Bit 7 **VSPOL:** Vertical synchronization polarity

This bit indicates the level on the VSYNC pin when the data are not valid on the parallel interface.

0: VSYNC active low

1: VSYNC active high

Bit 6 **HSPOL:** Horizontal synchronization polarity

This bit indicates the level on the HSYNC pin when the data are not valid on the parallel interface.

0: HSYNC active low

1: HSYNC active high

Bit 5 **PCKPOL:** Pixel clock polarity

This bit configures the capture edge of the pixel clock

0: Falling edge active.

1: Rising edge active.

Bit 4 ESS: Embedded synchronization select

- 0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the HSYNC/VSYNC signals.
- 1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.

Note: Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when the ESS bit is set.

This bit is disabled in JPEG mode.

Bit 3 JPEG: JPEG format

- 0: Uncompressed video format
- 1: This bit is used for JPEG data transfers. The HSYNC signal is used as data enable. The crop and embedded synchronization features (ESS bit) cannot be used in this mode.

Bits 2 CROP: Crop feature

- 0: The full image is captured. In this case the total number of bytes in an image frame should be a multiple of 4
- 1: Only the data inside the window specified by the crop register will be captured. If the size of the crop window exceeds the picture size, then only the picture size is captured.

Bit 1 CM: Capture mode

- 0: Continuous grab mode - The received data are transferred into the destination memory through the DMA. The buffer location and mode (linear or circular buffer) is controlled through the system DMA.
- 1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the DMA. At the end of the frame, the CAPTURE bit is automatically reset.

Bit 0 CAPTURE: Capture enable

- 0: Capture disabled.
- 1: Capture enabled.

The camera interface waits for the first start of frame, then a DMA request is generated to transfer the received data into the destination memory.

In snapshot mode, the CAPTURE bit is automatically cleared at the end of the 1st frame received.

In continuous grab mode, if the software clears this bit while a capture is ongoing, the bit will be effectively cleared after the frame end.

Note: The DMA controller and all DCMI configuration registers should be programmed correctly before enabling this bit.

12.8.2 DCMI status register (DCMI_SR)

Address offset: 0x04

Reset value: 0x0000 0x0000

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

FNE	VSYNC	HSYNC
r	r	r

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

Bit 2 **FNE:** FIFO not empty

This bit gives the status of the FIFO

1: FIFO contains valid data

0: FIFO empty

Bit 1 **VSYNC**

This bit gives the state of the VSYNC pin with the correct programmed polarity.

When embedded synchronization codes are used, the meaning of this bit is the following:

0: active frame

1: synchronization between frames

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

Bit 0 **Hsync**

This bit gives the state of the HSYNC pin with the correct programmed polarity.

When embedded synchronization codes are used, the meaning of this bit is the following:

0: active line

1: synchronization between lines

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

12.8.3 DCMI raw interrupt status register (DCMI_RIS)

Address offset: 0x08

Reset value: 0x0000 0x0000

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

LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
r	r	r	r	r

DCMI_RIS gives the raw interrupt status and is accessible in read only. When read, this register returns the status of the corresponding interrupt before masking with the DCMI_IER register value.

Bit 31:5 Reserved, must be kept at reset value.

Bit 4 LINE_RIS: Line raw interrupt status

This bit gets set when the HSYNC signal changes from the inactive state to the active state. It goes high even if the line is not valid.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMI_CR is set.

It is cleared by writing a '1' to the LINE_ISC bit in DCMI_ICR.

Bit 3 VSYNC_RIS: VSYNC raw interrupt status

This bit is set when the VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMI_CR.

It is cleared by writing a '1' to the VSYNC_ISC bit in DCMI_ICR.

Bit 2 ERR_RIS: Synchronization error raw interrupt status

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by writing a '1' to the ERR_ISC bit in DCMI_ICR.

Note: This bit is available only in embedded synchronization mode.

Bit 1 OVR_RIS: Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and the data FIFO is corrupted.

This bit is cleared by writing a '1' to the OVR_ISC bit in DCMI_ICR.

Bit 0 FRAME_RIS: Capture complete raw interrupt status

0: No new capture

1: A frame has been captured.

This bit is set when a frame or window has been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (e.g. window cropped outside the frame).

This bit is cleared by writing a '1' to the FRAME_ISC bit in DCMI_ICR.

12.8.4 DCMI interrupt enable register (DCMI_IER)

Address offset: 0x0C

Reset value: 0x0000 0x0000

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

LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
rw	rw	rw	rw	rw

The DCMI_IER register is used to enable interrupts. When one of the DCMI_IER bits is set, the corresponding interrupt is enabled. This register is accessible in both read and write.

Bit 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_IE:** Line interrupt enable

0: No interrupt generation when the line is received

1: An Interrupt is generated when a line has been completely received

Bit 3 **VSYNC_IE:** VSYNC interrupt enable

0: No interrupt generation

1: An interrupt is generated on each VSYNC transition from the inactive to the active state

The active state of the VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_IE:** Synchronization error interrupt enable

0: No interrupt generation

1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_IE:** Overrun interrupt enable

0: No interrupt generation

1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received.

Bit 0 **FRAME_IE:** Capture complete interrupt enable

0: No interrupt generation

1: An interrupt is generated at the end of each received frame/crop window (in crop mode).

12.8.5 DCMI masked interrupt status register (DCMI_MIS)

This DCMI_MIS register is a read-only register. When read, it returns the current masked status value (depending on the value in DCMI_IER) of the corresponding interrupt. A bit in this register is set if the corresponding enable bit in DCMI_IER is set and the corresponding bit in DCMI_RIS is set.

Address offset: 0x10

Reset value: 0x0000

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

LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
r	r	r	r	r

Bit 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_MIS:** Line masked interrupt status

This bit gives the status of the masked line interrupt

0: No interrupt generation when the line is received

1: An Interrupt is generated when a line has been completely received and the LINE_IE bit is set in DCMI_IER.

Bit 3 **VSYNC_MIS:** VSYNC masked interrupt status

This bit gives the status of the masked VSYNC interrupt

0: No interrupt is generated on VSYNC transitions

1: An interrupt is generated on each VSYNC transition from the inactive to the active state and the VSYNC_IE bit is set in DCMI_IER.

The active state of the VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_MIS:** Synchronization error masked interrupt status

This bit gives the status of the masked synchronization error interrupt

0: No interrupt is generated on a synchronization error

1: An interrupt is generated if the embedded synchronization codes are not received in the correct order and the ERR_IE bit in DCMI_IER is set.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_MIS:** Overrun masked interrupt status

This bit gives the status of the masked overflow interrupt

0: No interrupt is generated on overrun

1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received and the OVR_IE bit is set in DCMI_IER.

Bit 0 **FRAME_MIS:** Capture complete masked interrupt status

This bit gives the status of the masked capture complete interrupt

0: No interrupt is generated after a complete capture

1: An interrupt is generated at the end of each received frame/crop window (in crop mode) and the FRAME_IE bit is set in DCMI_IER.

12.8.6 DCMI interrupt clear register (DCMI_ICR)

Address offset: 0x14

Reset value: 0x0000 0x0000

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

LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
W	W	W	W	W

The DCMI_ICR register is write-only. Writing a ‘1’ into a bit of this register clears the corresponding bit in the DCMI_RIS and DCMI_MIS registers. Writing a ‘0’ has no effect.

Bit 15:5 Reserved, must be kept at reset value.

Bit 4 **LINE_ISC:** line interrupt status clear

Writing a ‘1’ into this bit clears LINE_RIS in the DCMI_RIS register

Bit 3 **VSYNC_ISC:** Vertical synch interrupt status clear

Writing a ‘1’ into this bit clears the VSYNC_RIS bit in DCMI_RIS

Bit 2 **ERR_ISC:** Synchronization error interrupt status clear

Writing a ‘1’ into this bit clears the ERR_RIS bit in DCMI_RIS

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_ISC:** Overrun interrupt status clear

Writing a ‘1’ into this bit clears the OVR_RIS bit in DCMI_RIS

Bits 0 **FRAME_ISC:** Capture complete interrupt status clear

Writing a ‘1’ into this bit clears the FRAME_RIS bit in DCMI_RIS

12.8.7 DCMI embedded synchronization code register (DCMI_ESCR)

Address offset: 0x18

Reset value: 0x0000 0x0000

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
FEC							LEC							LSC							FSC										
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		

Bit 31:24 **FEC**: Frame end delimiter code

This byte specifies the code of the frame end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FEC.

If FEC is programmed to 0xFF, all the unused codes (0xFF0000XY) are interpreted as frame end delimiters.

Bit 23:16 **LEC**: Line end delimiter code

This byte specifies the code of the line end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bit 15:8 **LSC**: Line start delimiter code

This byte specifies the code of the line start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bit 7:0 **FSC**: Frame start delimiter code

This byte specifies the code of the frame start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FSC.

If FSC is programmed to 0xFF, no frame start delimiter is detected. But, the 1st occurrence of LSC after an FEC code will be interpreted as a start of frame delimiter.

12.8.8 DCMI embedded synchronization unmask register (DCMI_ESUR)

Address offset: 0x1C

Reset value: 0x0000 0x0000

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
FEU								LEU								LSU								FSU							
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

Bit 31:24 FEU: Frame end delimiter unmask

This byte specifies the mask to be applied to the code of the frame end delimiter.

0: The corresponding bit in the FEC byte in DCMI_ESCR is masked while comparing the frame end delimiter with the received data.

1: The corresponding bit in the FEC byte in DCMI_ESCR is compared while comparing the frame end delimiter with the received data

Bit 23:16 LEU: Line end delimiter unmask

This byte specifies the mask to be applied to the code of the line end delimiter.

0: The corresponding bit in the LEC byte in DCMI_ESCR is masked while comparing the line end delimiter with the received data

1: The corresponding bit in the LEC byte in DCMI_ESCR is compared while comparing the line end delimiter with the received data

Bit 15:8 LSU: Line start delimiter unmask

This byte specifies the mask to be applied to the code of the line start delimiter.

0: The corresponding bit in the LSC byte in DCMI_ESCR is masked while comparing the line start delimiter with the received data

1: The corresponding bit in the LSC byte in DCMI_ESCR is compared while comparing the line start delimiter with the received data

Bit 7:0 FSU: Frame start delimiter unmask

This byte specifies the mask to be applied to the code of the frame start delimiter.

0: The corresponding bit in the FSC byte in DCMI_ESCR is masked while comparing the frame start delimiter with the received data

1: The corresponding bit in the FSC byte in DCMI_ESCR is compared while comparing the frame start delimiter with the received data

12.8.9 DCMI crop window start (DCMI_CWSTART)

Address offset: 0x20

Reset value: 0x0000 0x0000

	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	VST[12:0]														Reserved	HOFFCNT[13:0]																
	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 31:29 Reserved, must be kept at reset value.

Bit 28:16 **VST[12:0]:** Vertical start line count

The image capture starts with this line number. Previous line data are ignored.

0x0000 => line 1

0x0001 => line 2

0x0002 => line 3

....

Bits 15:14 Reserved, must be kept at reset value.

Bit 13:0 **HOFFCNT[13:0]:** Horizontal offset count

This value gives the number of pixel clocks to count before starting a capture.

12.8.10 DCMI crop window size (DCMI_CWSIZE)

Address offset: 0x24

Reset value: 0x0000 0x0000

	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	VLINE[13:0]														Reserved	CAPCNT[13:0]																
	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.

Bit 29:16 **VLINE[13:0]:** Vertical line count

This value gives the number of lines to be captured from the starting point.

0x0000 => 1 line

0x0001 => 2 lines

0x0002 => 3 lines

....

Bits 15:14 Reserved, must be kept at reset value.

Bit 13:0 **CAPCNT[13:0]:** Capture count

This value gives the number of pixel clocks to be captured from the starting point on the same line. It value should corresponds to word-aligned data for different widths of parallel interfaces.

0x0000 => 1 pixel

0x0001 => 2 pixels

0x0002 => 3 pixels

....

12.8.11 DCMI data register (DCMI_DR)

Address offset: 0x28

Reset value: 0x0000 0x0000

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	
Byte3						Byte2						Byte1						Byte0														
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r		

Bits 31:24 Data byte 3

Bit 23:16 Data byte 2

Bits 15:8 Data byte 1

Bit 7:0 Data byte 0

The digital camera Interface packages all the received data in 32-bit format before requesting a DMA transfer. A 4-word deep FIFO is available to leave enough time for DMA transfers and avoid DMA overrun conditions.

12.8.12 DCMI register map

Table 54 summarizes the DCMI registers.

Table 54. DCMI 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	DCMI_CR	Reserved														ENABLE 0	EDM		FCRC		VSPOL		HSPOL		PCKPOL		5		ESS		4		JPEG		3		2		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	DCMI_SR	Reserved														FNE		CROP		CM		VSYNC		0		HSYNC		0		0		0		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								
0x08	DCMI_RIS	Reserved														LINE_RIS		VSYNC_RIS		ERR_RIS		OVR_RIS		FRAME_RIS		LINE_ISC		VSYNC_ISC		ERR_ISC		OVR_ISC		FRAME_ISC		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								
0x0C	DCMI_IER	Reserved														LINE_IE		VSYNC_IE		ERR_IE		OVR_IE		FRAME_IE		LINE_ISC		VSYNC_ISC		ERR_ISC		OVR_ISC		FRAME_ISC		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								
0x10	DCMI_MIS	Reserved														LINE_MIS		VSYNC_MIS		ERR_MIS		OVR_MIS		FRAME_MIS		LINE_ISC		VSYNC_ISC		ERR_ISC		OVR_ISC		FRAME_ISC		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								
0x14	DCMI_ICR	Reserved														LINE_RIS		VSYNC_RIS		ERR_RIS		OVR_RIS		FRAME_RIS		LINE_ISC		VSYNC_ISC		ERR_ISC		OVR_ISC		FRAME_ISC		0						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0		0		0		0		0		0		0		0		0								

Table 54. DCMI register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	DCMI_ESCR	FEC				LEC				LSC				FSC																			
		0	0	0	0	0	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	DCMI_ESUR	FEU				LEU				LSU				FSU																			
		0	0	0	0	0	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	DCMI_CWSTRT	Reserved	VST[12:0]												Reserved	HOFFCNT[13:0]																	
			0	0	0	0	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	DCMI_CWSIZE	Reserved	VLINE13:0]												Reserved	CAPCNT[13:0]																	
			0	0	0	0	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	DCMI_DR	Byte3				Byte2				Byte1				Byte0																			
		0	0	0	0	0	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 [Table 1 on page 50](#) for the register boundary addresses.

13 Advanced-control timers (TIM1&TIM8)

13.1 TIM1&TIM8 introduction

The advanced-control timers (TIM1&TIM8) consist of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

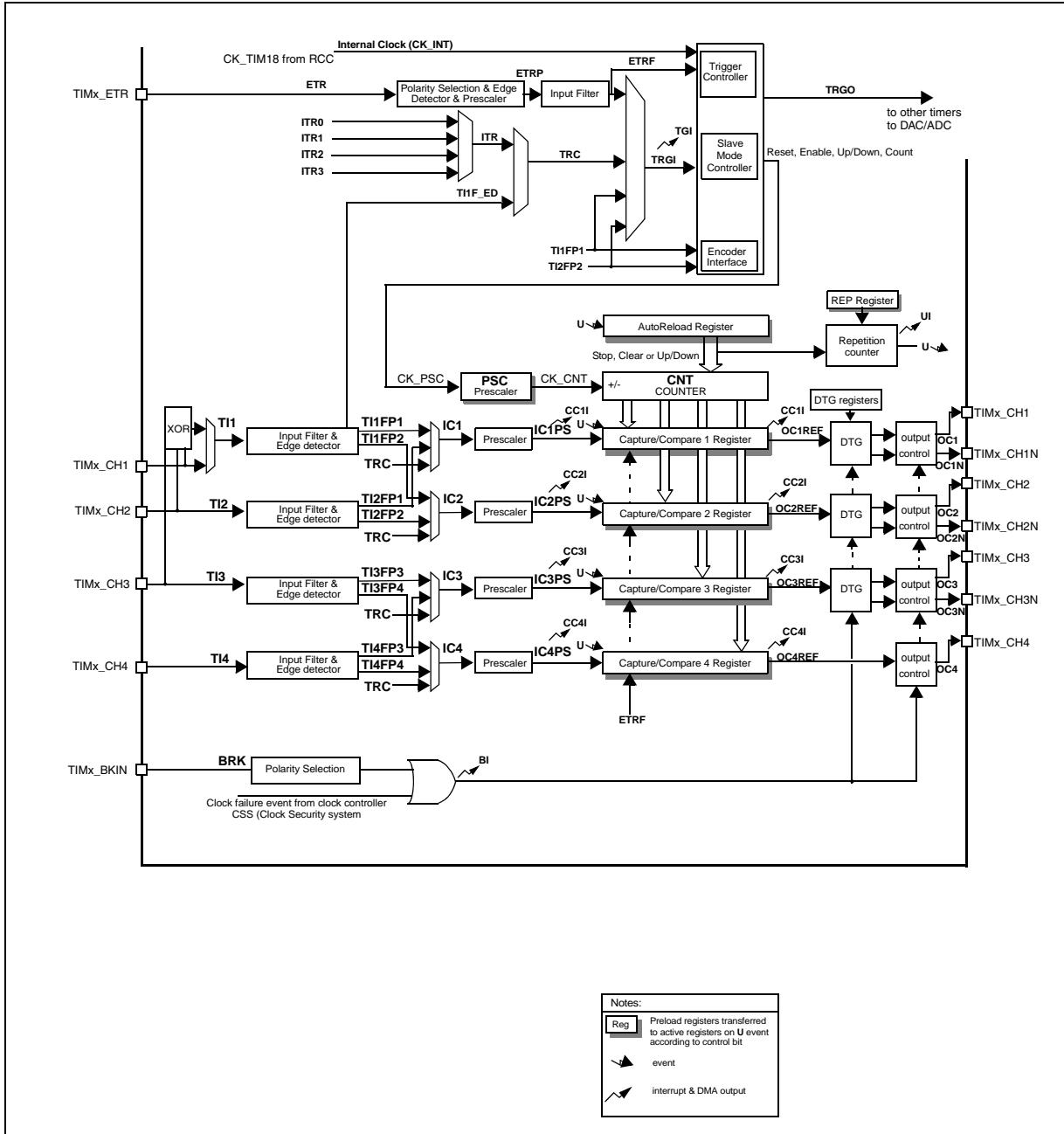
The advanced-control (TIM1&TIM8) and general-purpose (TIMx) timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 13.3.20](#).

13.2 TIM1&TIM8 main features

TIM1&TIM8 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65535.
- Up to 4 independent channels for:
 - Input Capture
 - Output Compare
 - PWM generation (Edge and Center-aligned Mode)
 - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- Break input to put the timer’s output signals in reset state or in a known state.
- Interrupt/DMA generation on the following events:
 - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
 - Trigger event (counter start, stop, initialization or count by internal/external trigger)
 - Input capture
 - Output compare
 - Break input
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 65. Advanced-control timer block diagram



13.3 TIM1&TIM8 functional description

13.3.1 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 detail for each configuration.

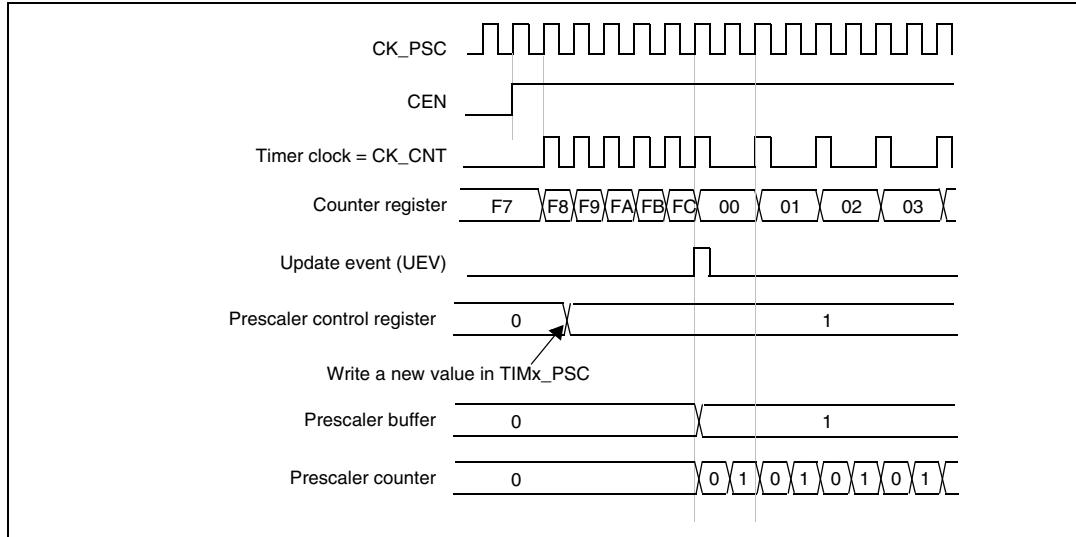
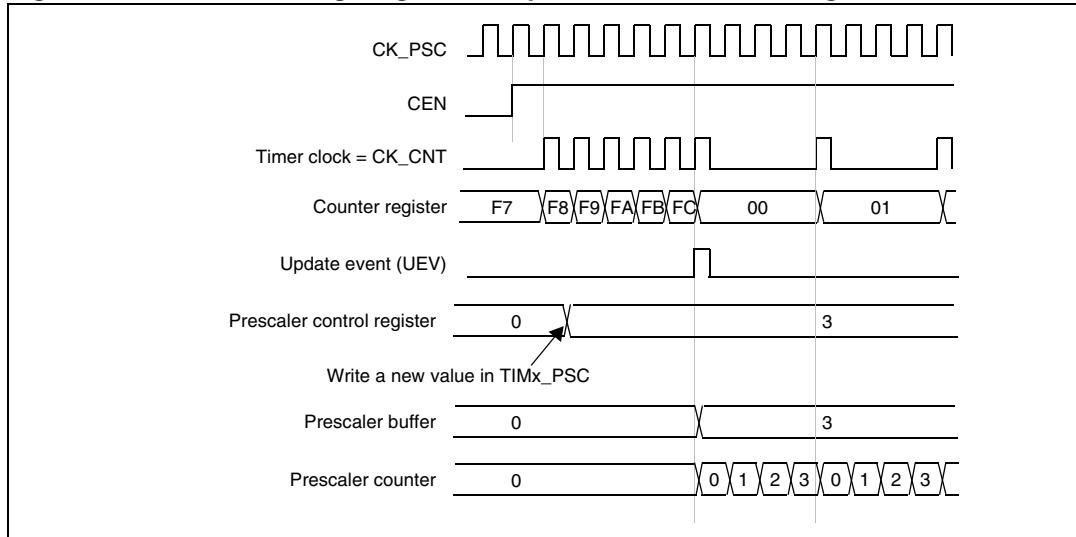
The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the 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 67 and *Figure 68* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 66. Counter timing diagram with prescaler division change from 1 to 2**Figure 67. Counter timing diagram with prescaler division change from 1 to 4**

13.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

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 68. Counter timing diagram, internal clock divided by 1

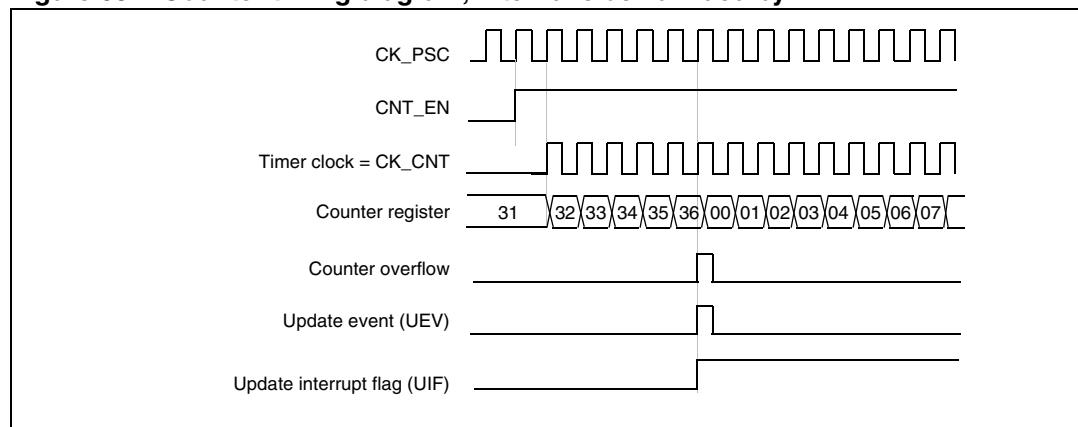


Figure 69. Counter timing diagram, internal clock divided by 2

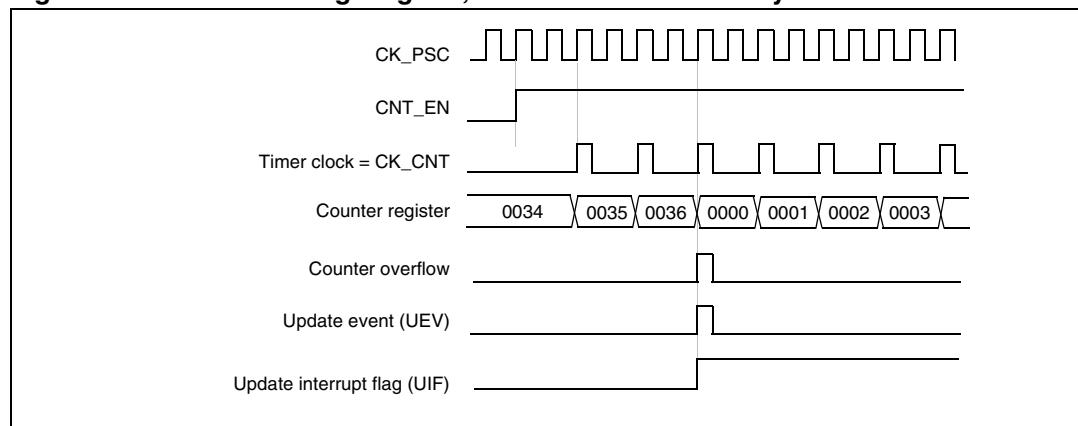
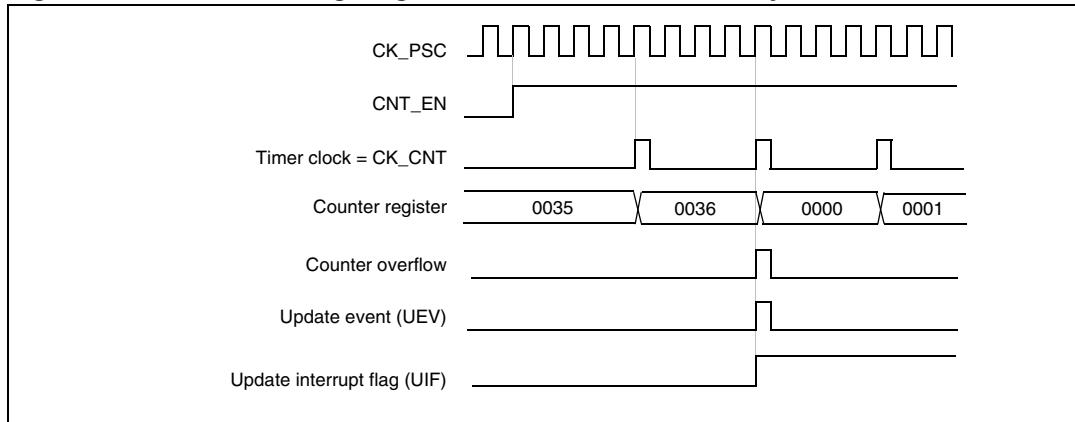
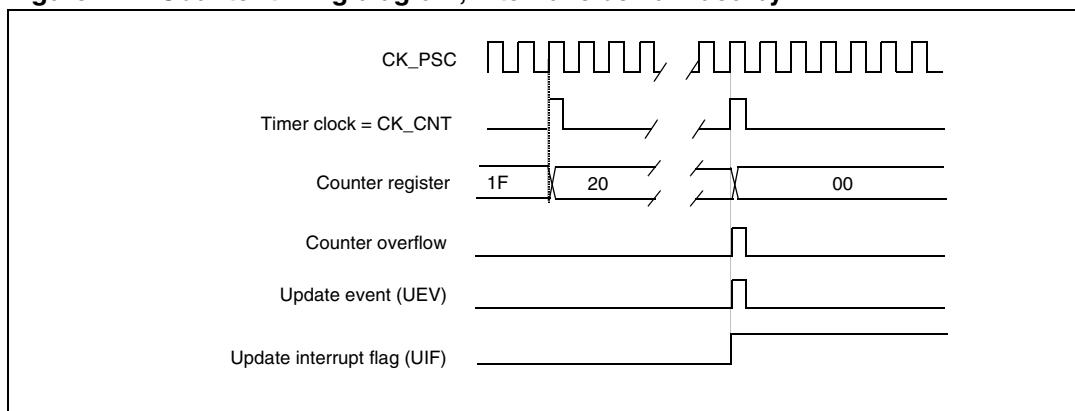
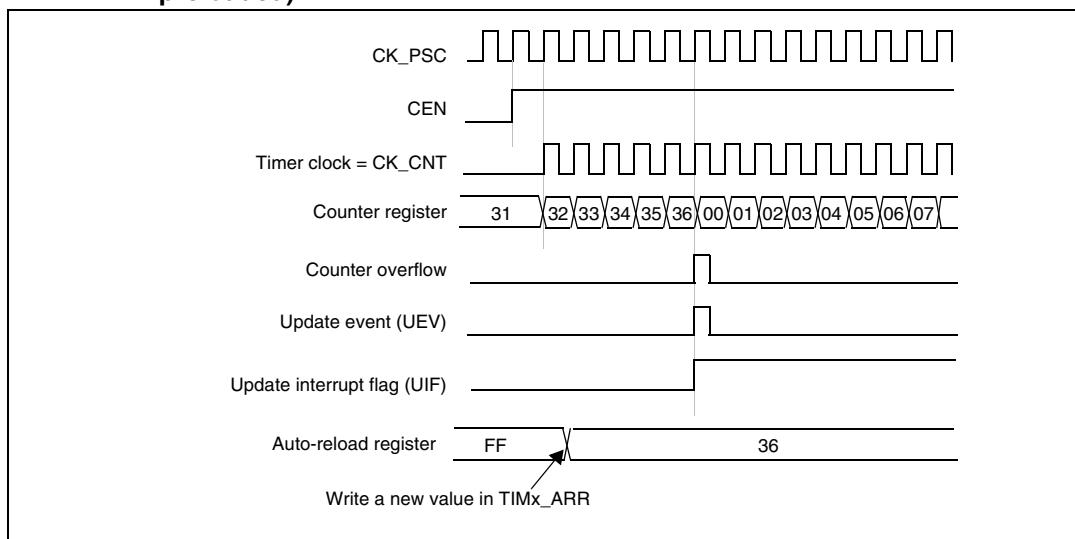
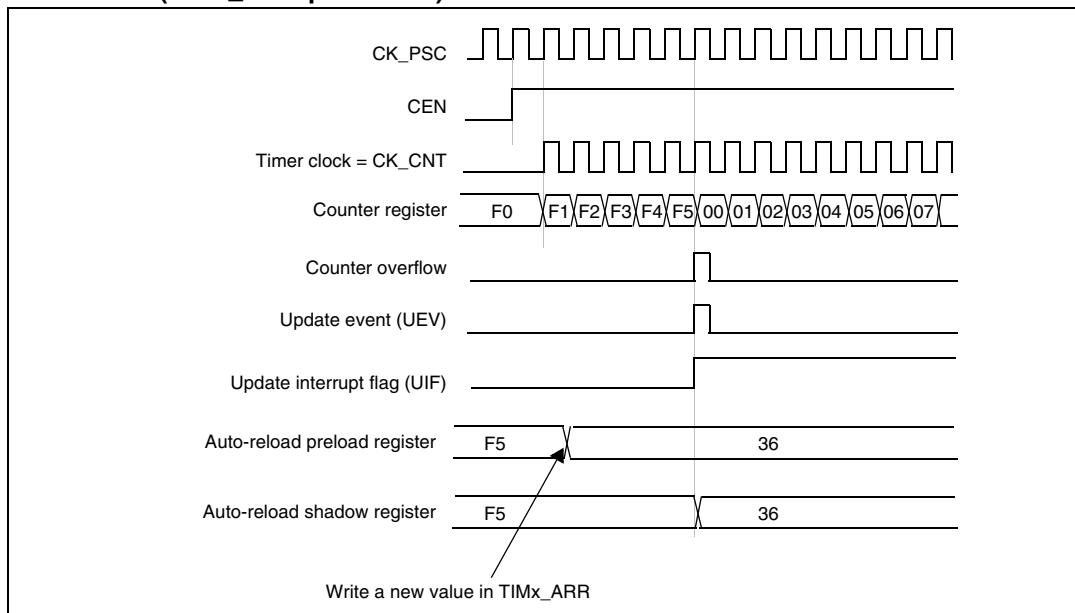


Figure 70. Counter timing diagram, internal clock divided by 4**Figure 71. Counter timing diagram, internal clock divided by N****Figure 72. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)**

**Figure 73. Counter timing diagram, update event when ARPE=1
(TIMx_ARR preloaded)**



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 74. Counter timing diagram, internal clock divided by 1

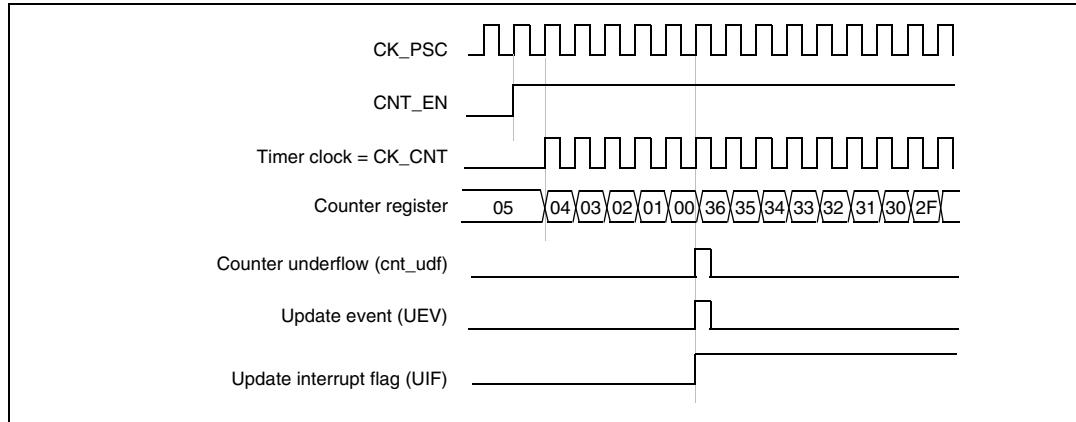


Figure 75. Counter timing diagram, internal clock divided by 2

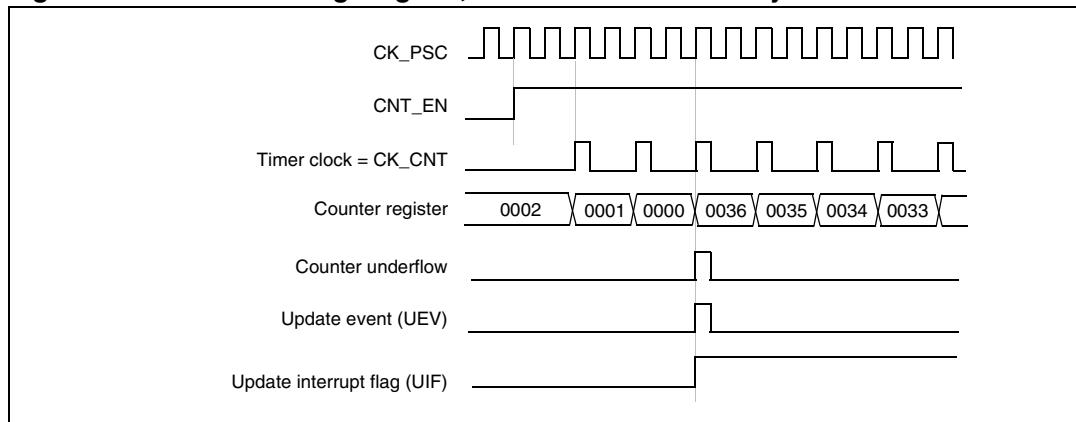


Figure 76. Counter timing diagram, internal clock divided by 4

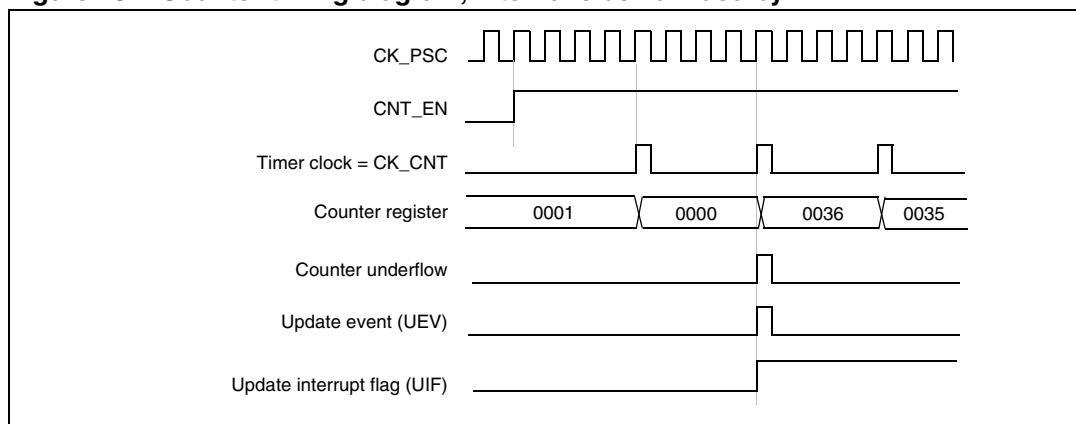
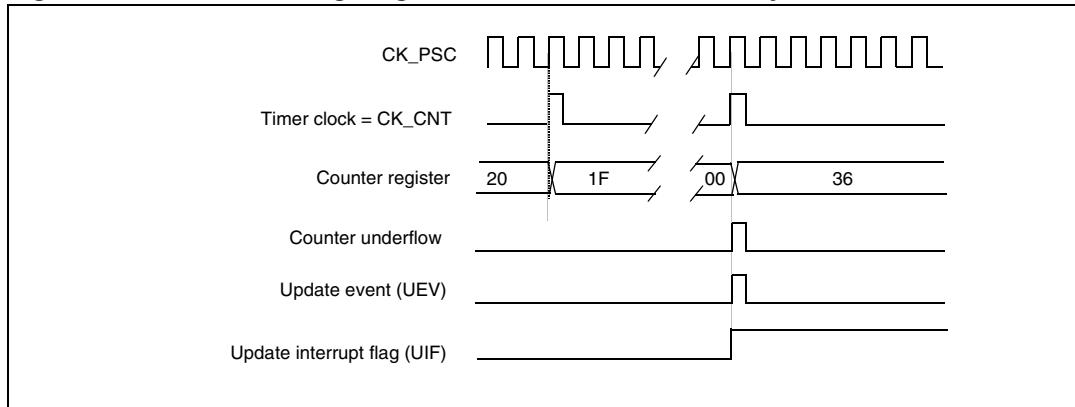
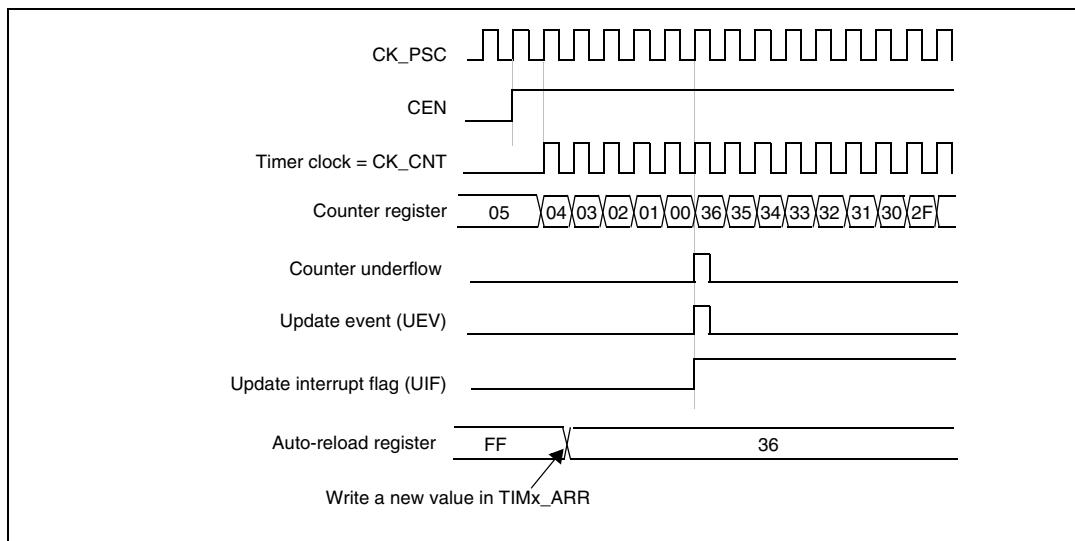


Figure 77. Counter timing diagram, internal clock divided by N**Figure 78. Counter timing diagram, update event when repetition counter is not used**

Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") or the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

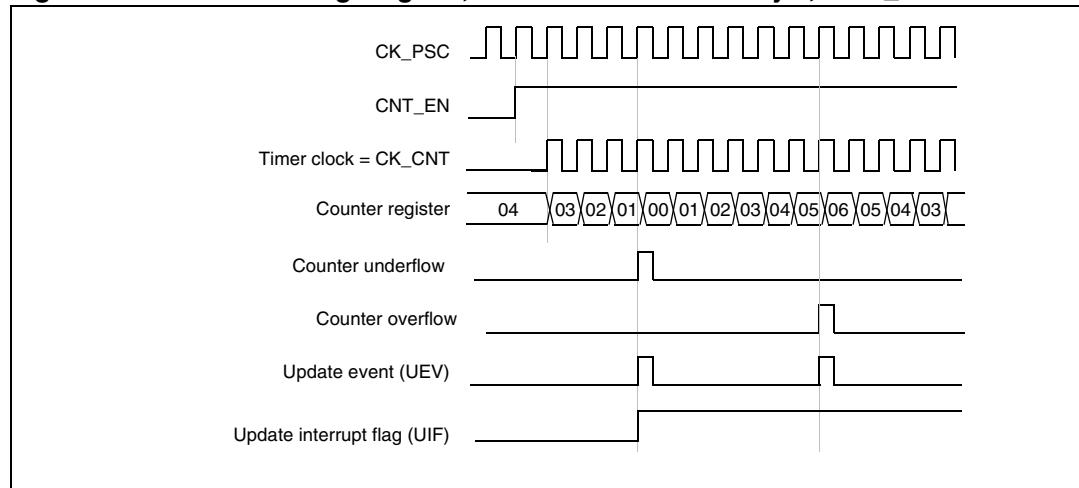
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

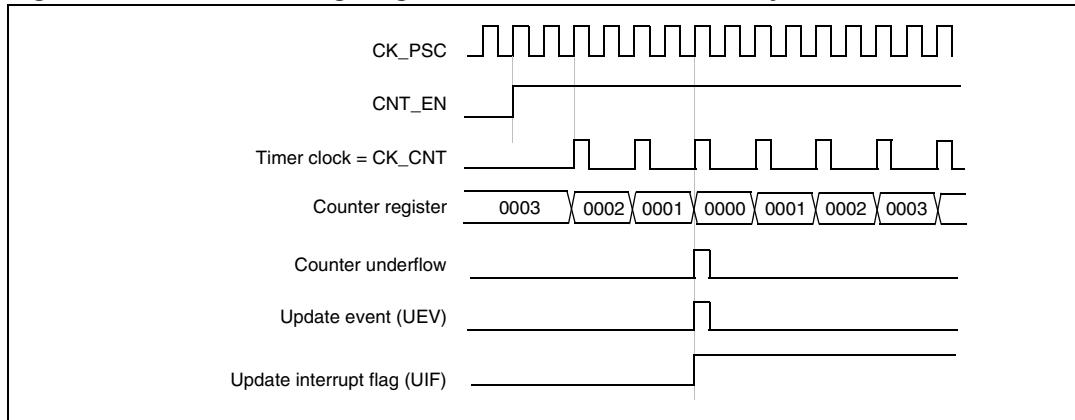
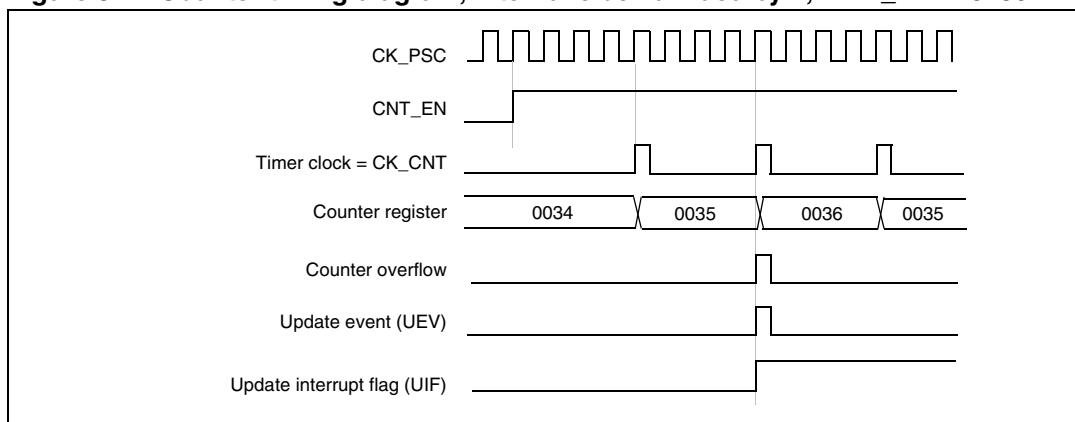
- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 79. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to [Section 13.4: TIM1&TIM8 registers on page 333](#)).

Figure 80. Counter timing diagram, internal clock divided by 2**Figure 81. 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.

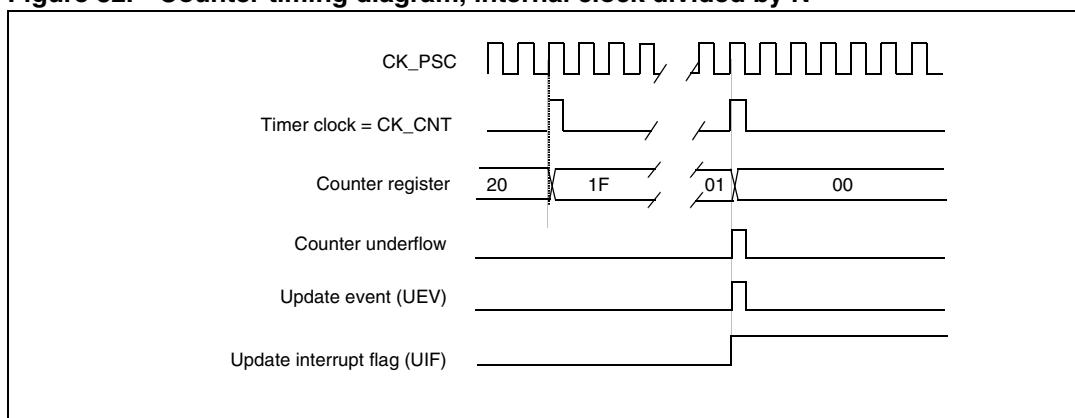
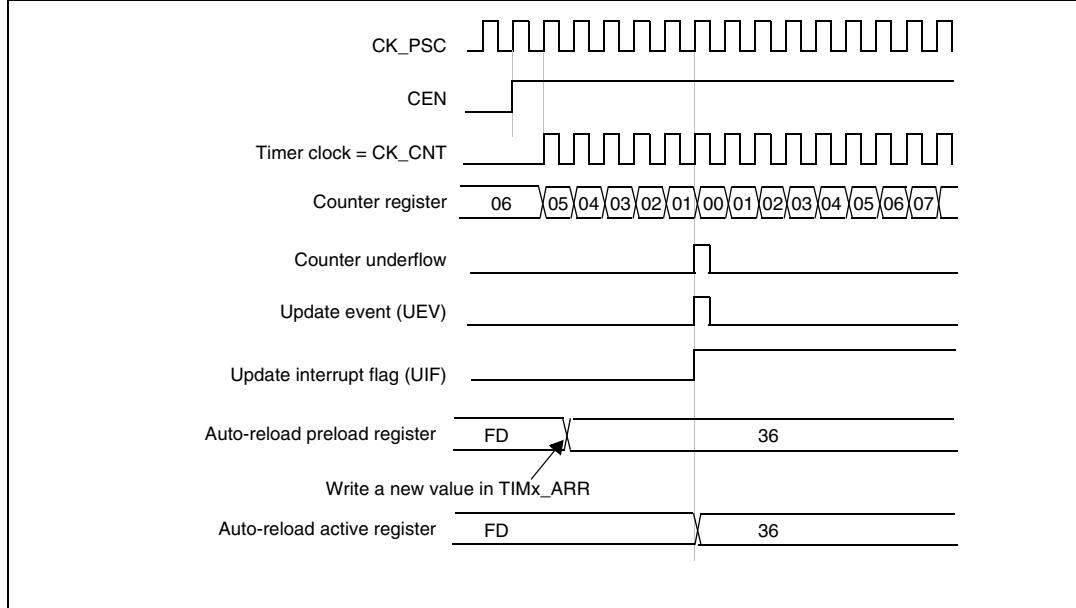
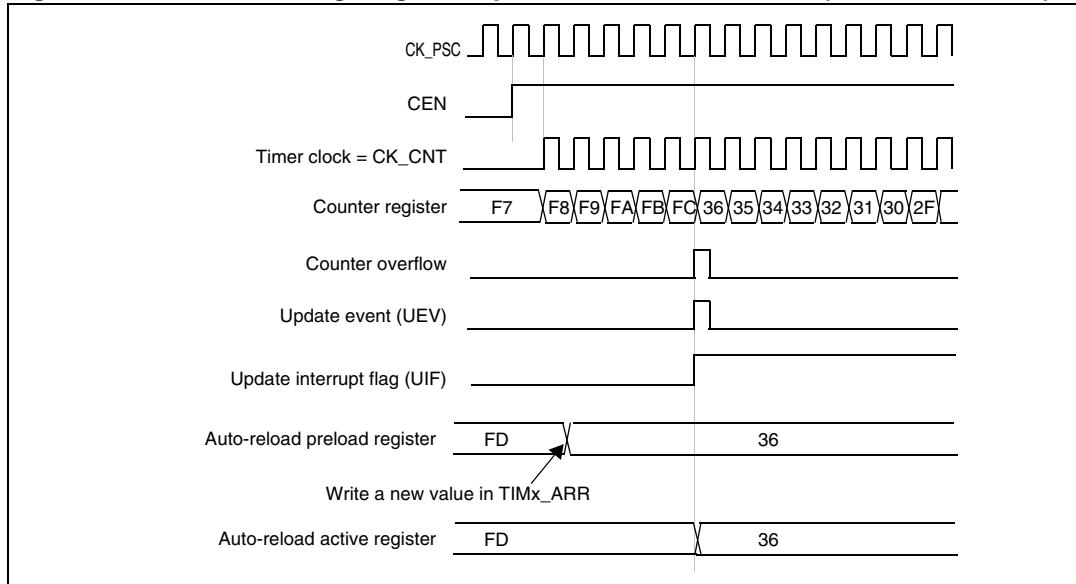
Figure 82. Counter timing diagram, internal clock divided by N

Figure 83. Counter timing diagram, update event with ARPE=1 (counter underflow)**Figure 84. Counter timing diagram, Update event with ARPE=1 (counter overflow)**

13.3.3 Repetition counter

[Section 13.3.1: Time-base unit](#) describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

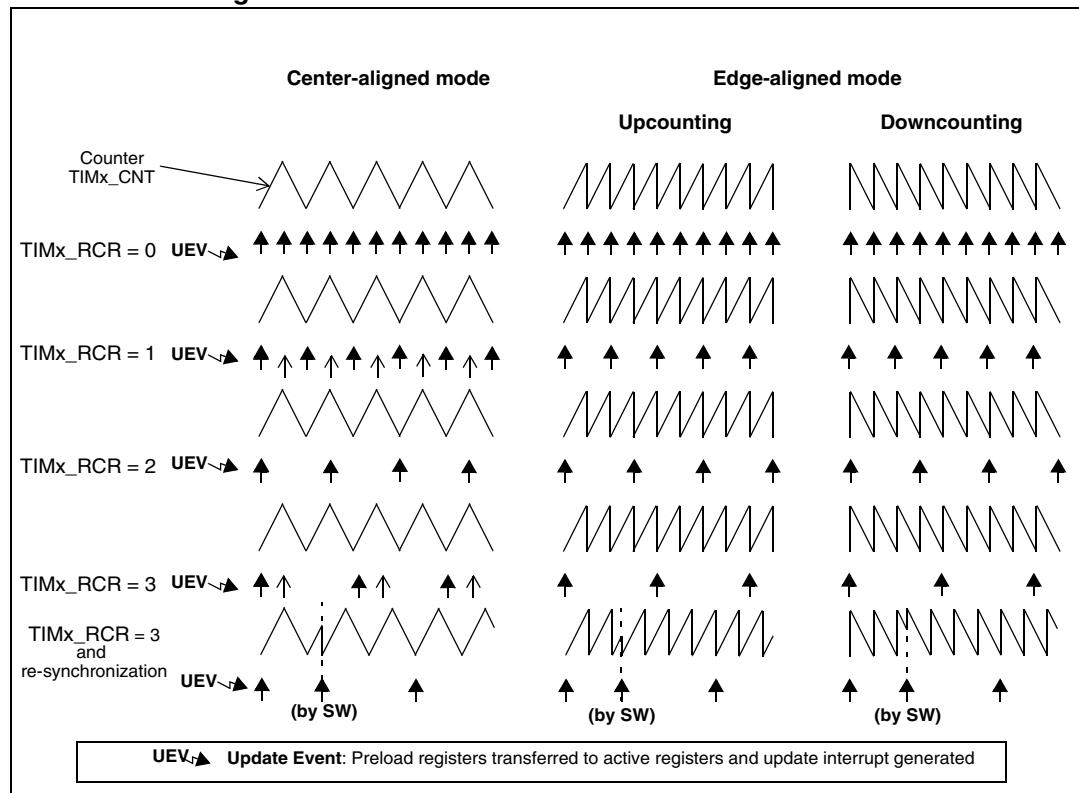
The repetition counter is decremented:

- At each counter overflow in upcounting mode,
 - At each counter underflow in downcounting mode,
 - At each counter overflow and at each counter underflow in center-aligned mode.
- Although this limits the maximum number of repetition to 128 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2 \times T_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to [Figure 85](#)). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In center-aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was started. If the RCR was written before starting the counter, the UEV occurs on the overflow. If the RCR was written after starting the counter, the UEV occurs on the underflow. For example for RCR = 3, the UEV is generated on each 4th overflow or underflow event depending on when RCR was written.

Figure 85. Update rate examples depending on mode and TIMx_RCR register settings



13.3.4 Clock selection

The counter clock can be provided by the following clock sources:

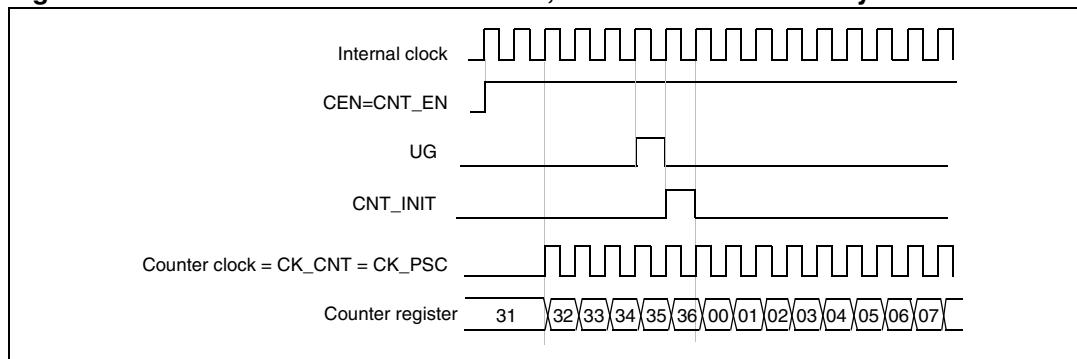
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, you can configure Timer 1 to act as a prescaler for Timer 2. Refer to [Using one timer as prescaler for another](#) for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

[Figure 86](#) shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

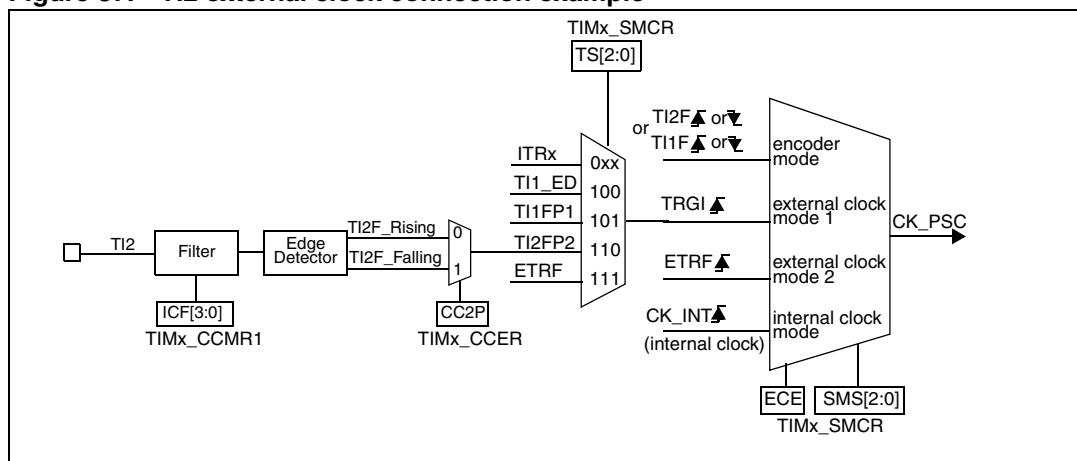
Figure 86. Control circuit in normal mode, internal clock divided by 1



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 87. 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:

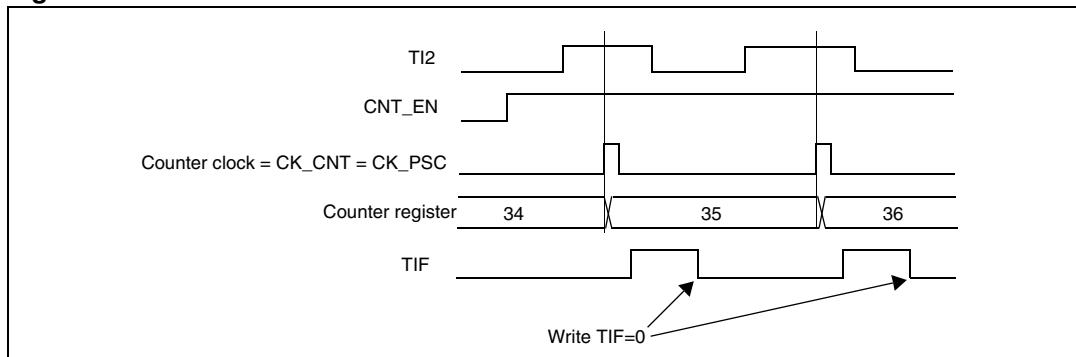
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 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 88. Control circuit in external clock mode 1



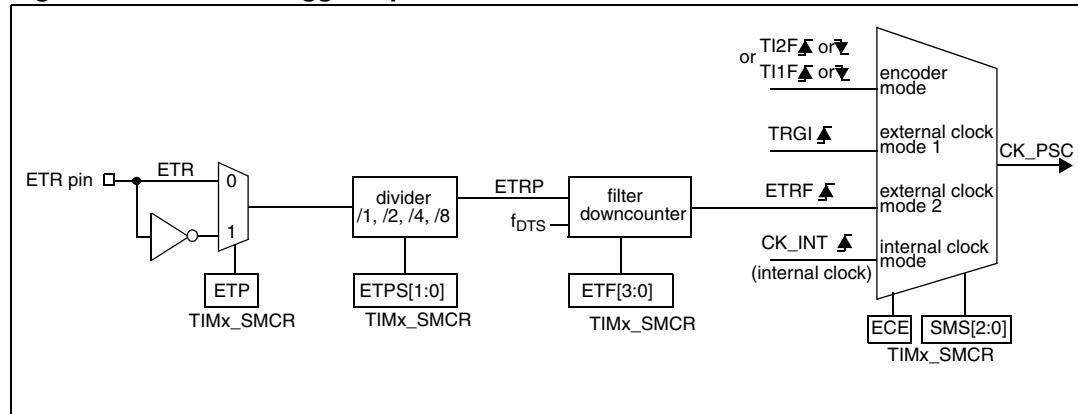
External clock source mode 2

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 89 gives an overview of the external trigger input block.

Figure 89. External trigger input block



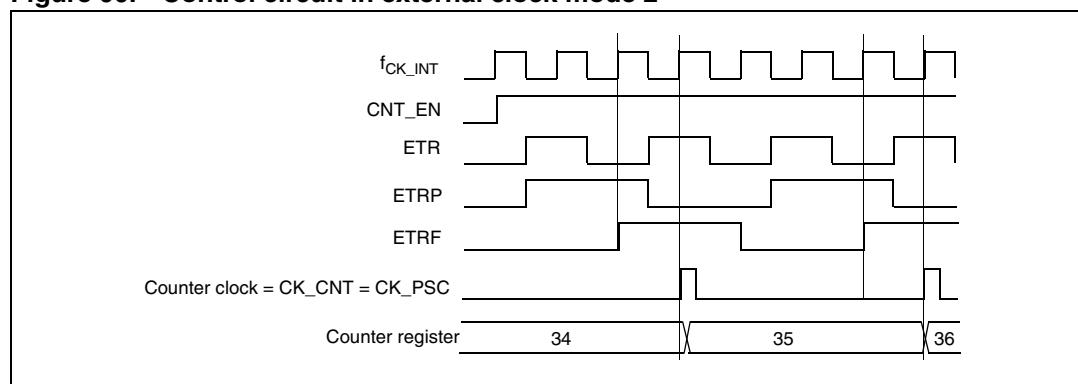
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

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 90. Control circuit in external clock mode 2



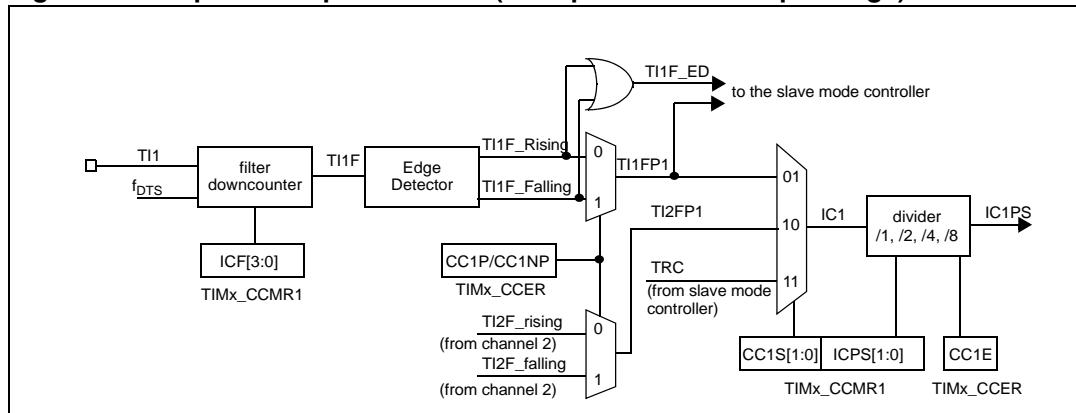
13.3.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 91 to *Figure 94* 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 91. 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 92. Capture/compare channel 1 main circuit

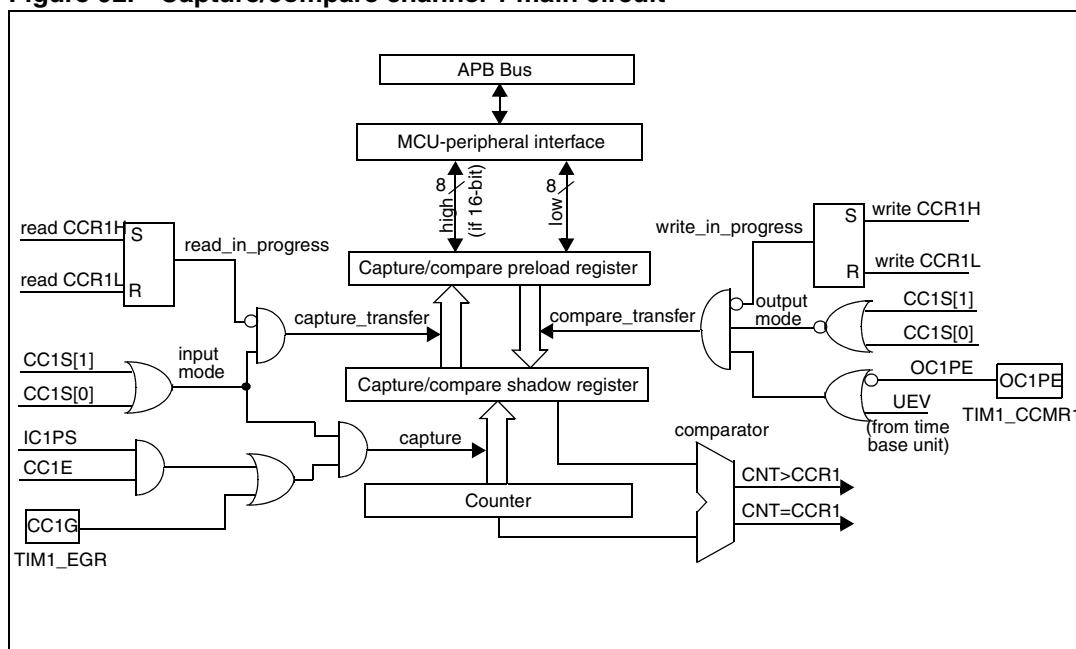
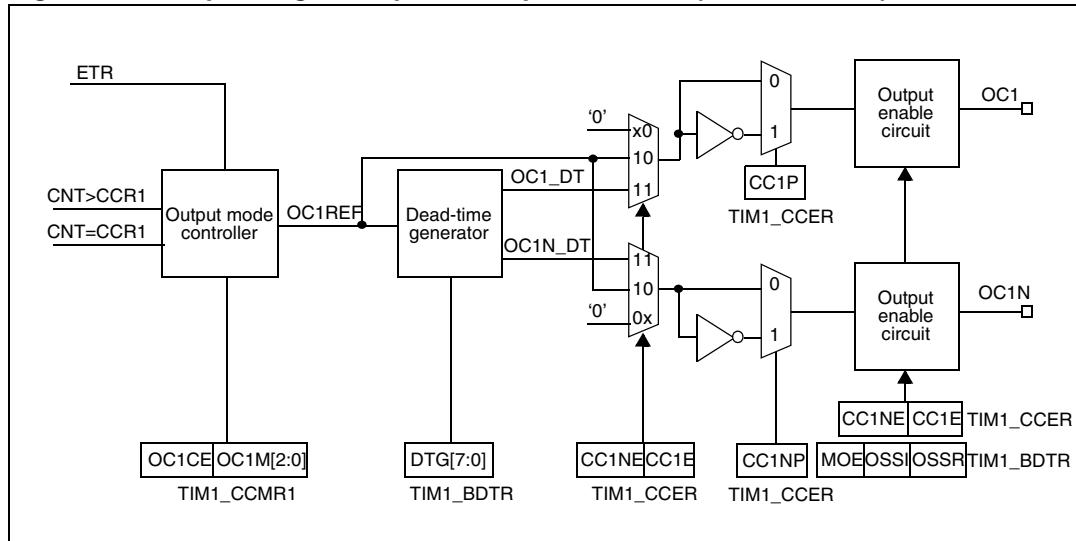
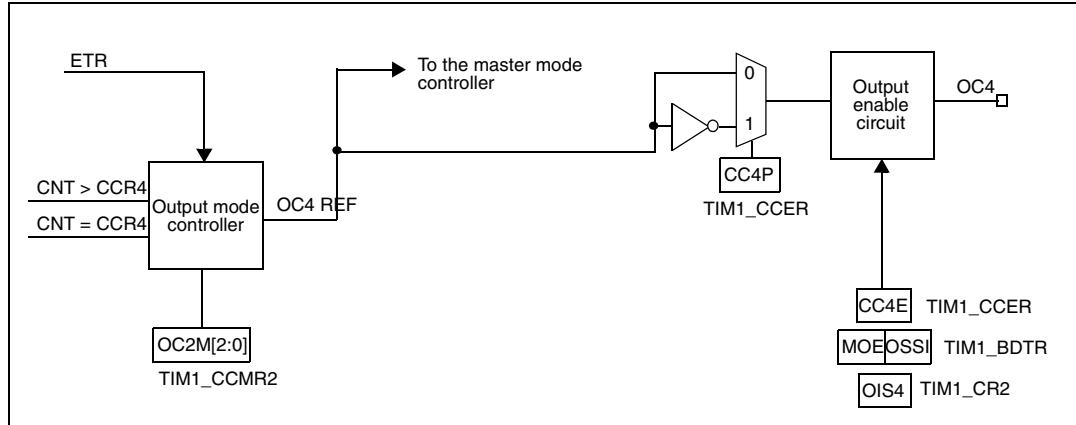


Figure 93. Output stage of capture/compare channel (channel 1 to 3)**Figure 94. Output stage of capture/compare channel (channel 4)**

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

13.3.6 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.

13.3.7 PWM input mode

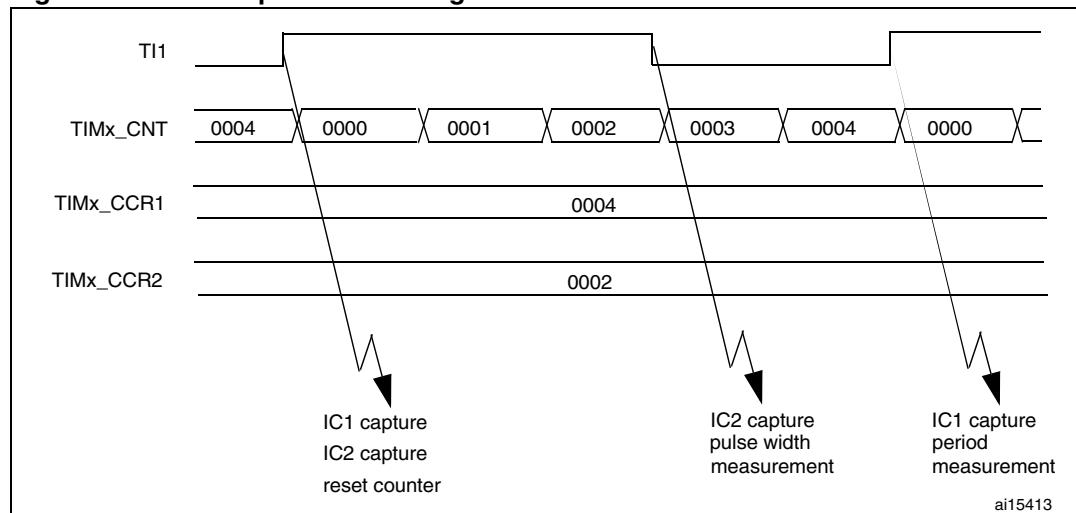
This mode is a particular case of input capture mode. The procedure is the same except:

- 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 '1' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 95. PWM input mode timing



13.3.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

13.3.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCXM=001), be set inactive (OCXM=010) or can toggle (OCXM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

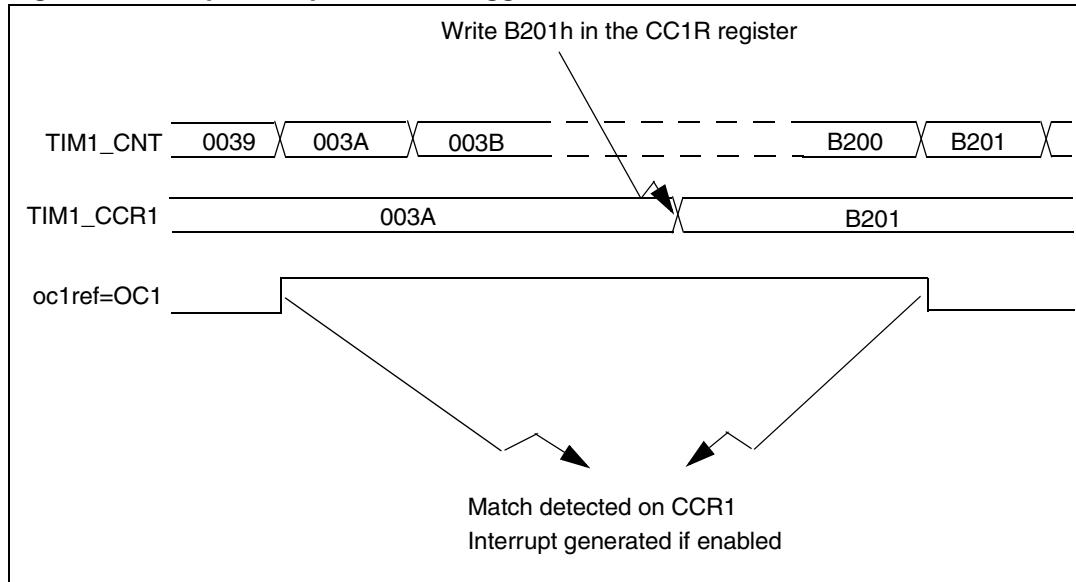
The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
 - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
 - Write OCxPE = 0 to disable preload register
 - Write CCxP = 0 to select active high polarity
 - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 96](#).

Figure 96. Output compare mode, toggle on OC1.

13.3.10 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘110’ (PWM mode 1) or ‘111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSS1 and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx \leq TIMx_CNT or TIMx_CNT \leq TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

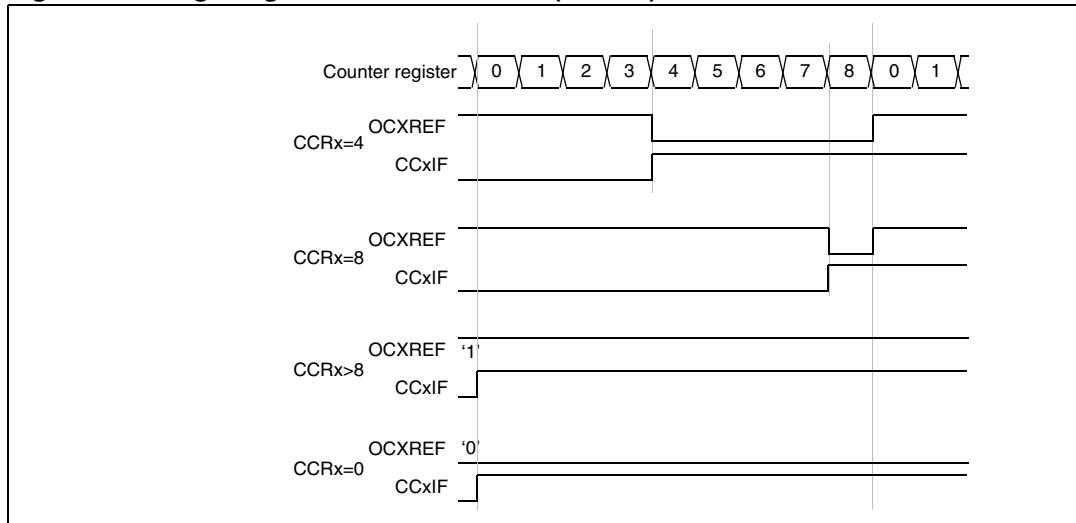
- Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to [Section : Upcounting mode on page 296](#).

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as $\text{TIMx_CNT} < \text{TIMx_CCR}_x$ else it becomes low. If the compare value in TIMx_CCR_x 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 97](#) shows some edge-aligned PWM waveforms in an example where $\text{TIMx_ARR}=8$.

Figure 97. Edge-aligned PWM waveforms (ARR=8)



- Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to [Section : Downcounting mode on page 299](#)

In PWM mode 1, the reference signal OCxRef is low as long as $\text{TIMx_CNT} > \text{TIMx_CCR}_x$ else it becomes high. If the compare value in TIMx_CCR_x is greater than the auto-reload value in TIMx_ARR , then OCxREF is held at '1'. 0% PWM is not possible in this mode.

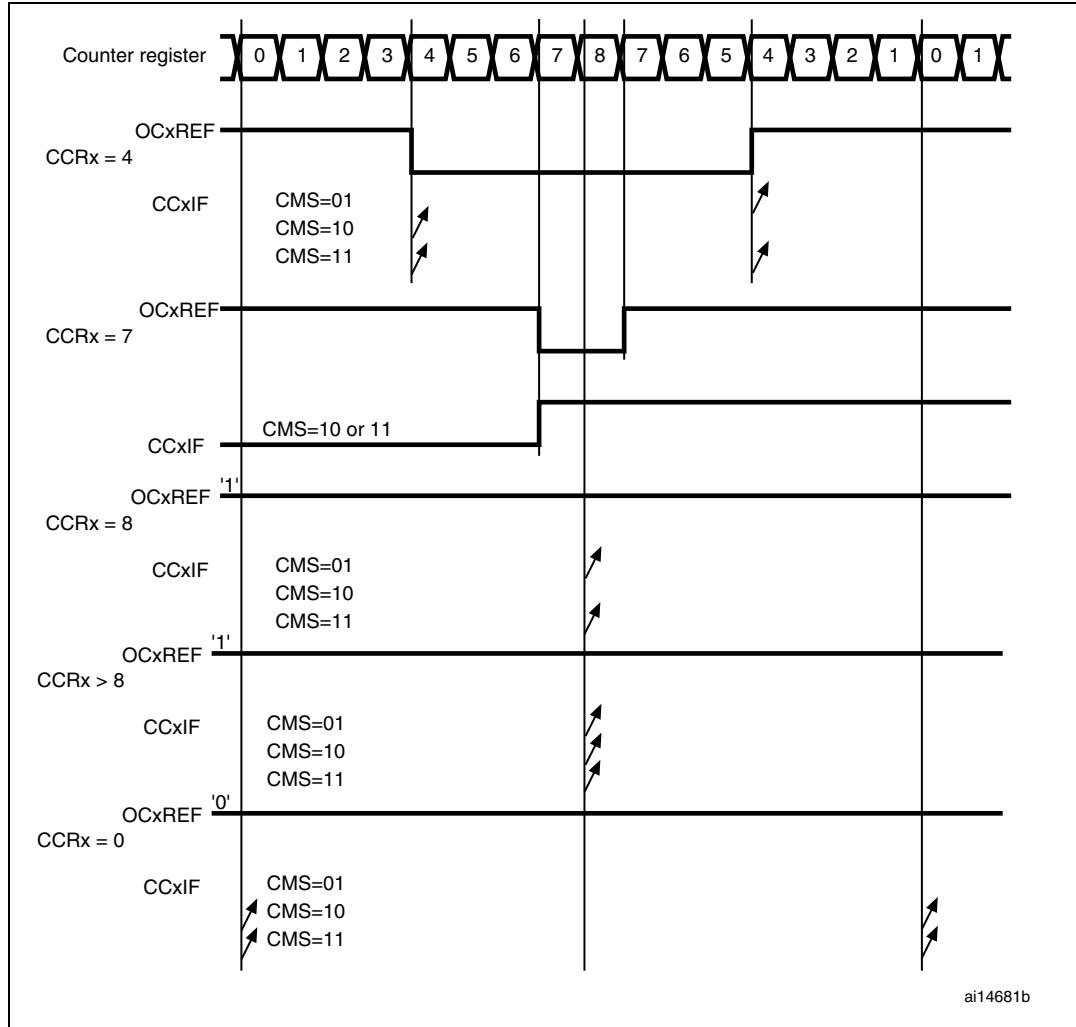
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to [Section : Center-aligned mode \(up/down counting\) on page 301](#).

[Figure 98](#) shows some center-aligned PWM waveforms in an example where:

- $\text{TIMx_ARR}=8$,
- 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 98. Center-aligned PWM waveforms (ARR=8)

**Hints on using center-aligned mode:**

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
 - The direction is not updated if you write a value in the counter that is greater than the auto-reload value (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
 - The direction is updated if you write 0 or write the TIMx_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

13.3.11 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1&TIM8) 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 OC_x or complementary OC_{xN}) independently for each output. This is done by writing to the CC_{xP} and CC_{xNP} bits in the TIM_x_CCER register.

The complementary signals OC_x and OC_{xN} are activated by a combination of several control bits: the CC_{xE} and CC_{xNE} bits in the TIM_x_CCER register and the MOE, OIS_x, OIS_{xN}, OSSI and OSSR bits in the TIM_x_BDTR and TIM_x_CR2 registers. Refer to [Table 57: Output control bits for complementary OC_x and OC_{xN} channels with break feature on page 351](#) for more details. In particular, the dead-time is activated when switching to the IDLE state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CC_{xE} and CC_{xNE} bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OC_{xREF}, it generates 2 outputs OC_x and OC_{xN}. If OC_x and OC_{xN} are active high:

- The OC_x output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OC_{xN} output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OC_x or OC_{xN}) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OC_{xREF}. (we suppose CC_{xP}=0, CC_{xNP}=0, MOE=1, CC_{xE}=1 and CC_{xNE}=1 in these examples)

Figure 99. Complementary output with dead-time insertion.

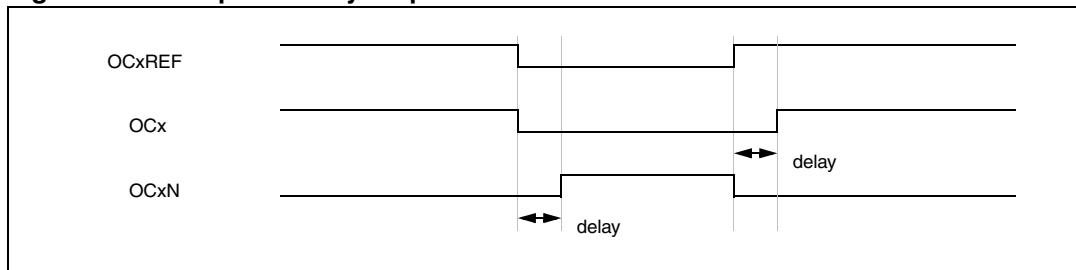


Figure 100. Dead-time waveforms with delay greater than the negative pulse.

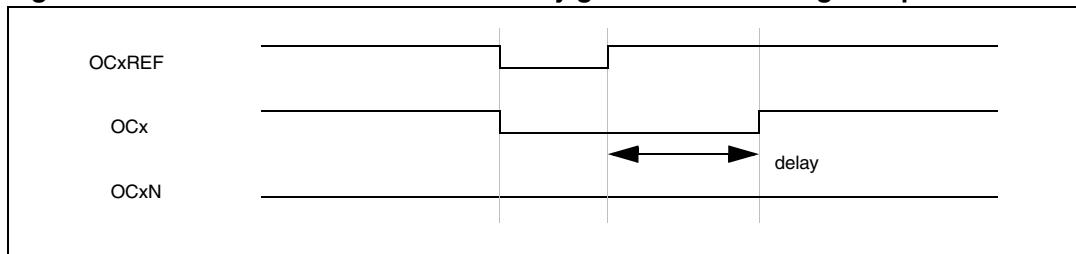
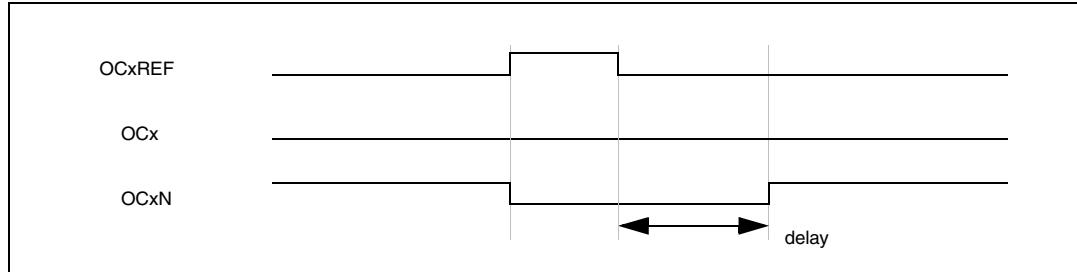


Figure 101. Dead-time waveforms with delay greater than the positive pulse.

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 13.4.18: TIM1&TIM8 break and dead-time register \(TIMx_BDTR\) on page 355](#) 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.

13.3.12 Using the break function

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSS1 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 57: Output control bits for complementary OCx and OCxN channels with break feature on page 351](#) for more details.

The break source can be either the break input pin or a clock failure event, generated by the Clock Security System (CSS), from the Reset Clock Controller. For further information on the Clock Security System, refer to [Section 5.2.7: Clock security system \(CSS\)](#).

When exiting from reset, the break circuit is disabled and the MOE bit is low. You can enable the break function by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if you write MOE to 1 whereas it was low, you

must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read the synchronous signal.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or in reset state (selected by the 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 then the timer releases the enable output else the enable output remains high.
- When complementary outputs are used:
 - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
 - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).
 - If OSS1=0 then the timer releases the enable outputs else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until you write it to ‘1’ again. In this case, it can be used for security and you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

Note:

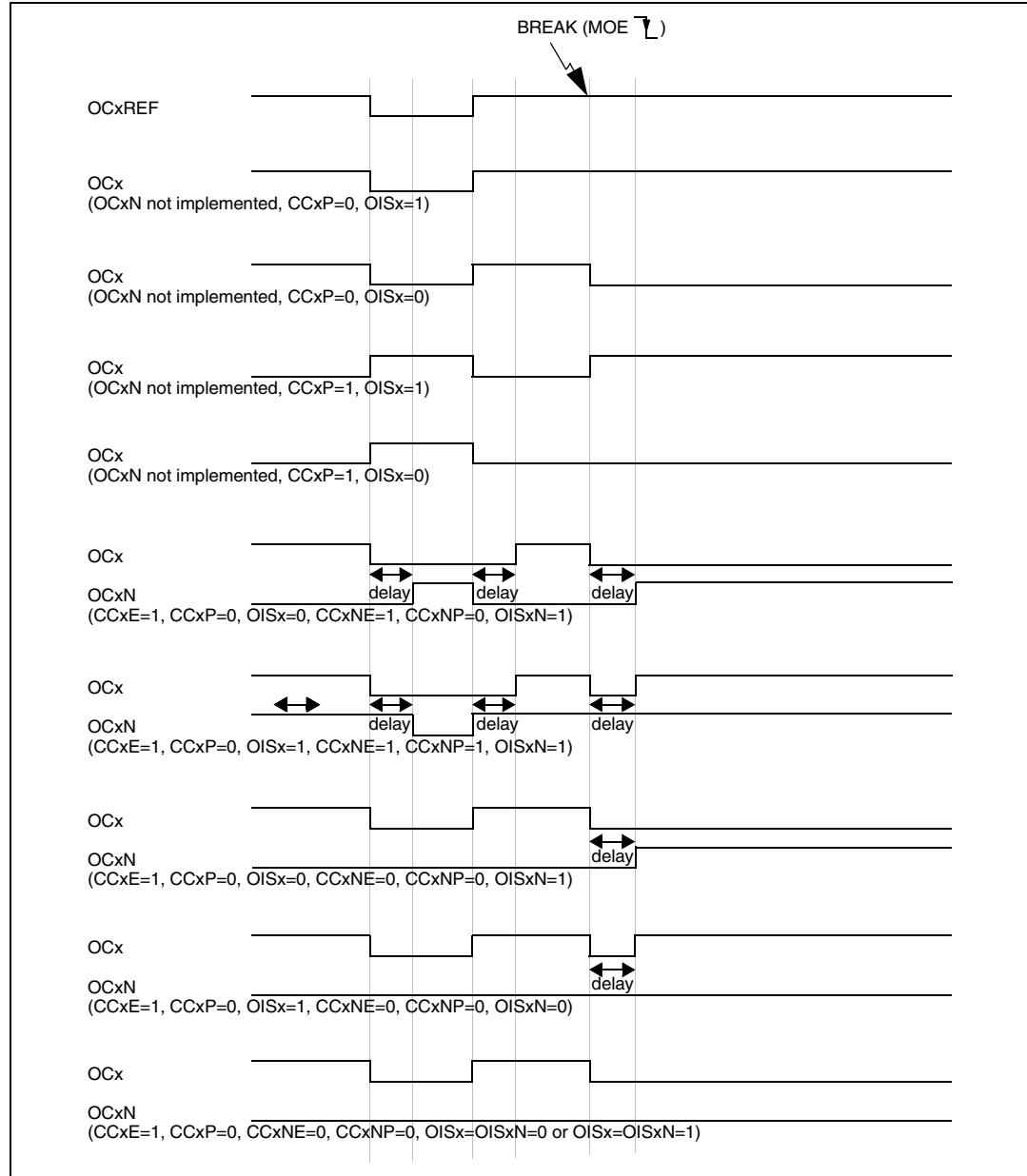
The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows you to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). You can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to [Section 13.4.18: TIM1&TIM8 break and dead-time register \(TIMx_BDTR\) on page 355](#). The LOCK bits can be written only once after an MCU reset.

[Figure 102](#) shows an example of behavior of the outputs in response to a break.

Figure 102. Output behavior in response to a break.



13.3.13 Clearing the OCxREF signal on an external event

The OC_xREF signal for a given channel can be driven Low by applying a High level to the ETRF input (OC_xCE enable bit of the corresponding TIM_x_CCMR_x register set to '1'). The OC_xREF signal remains Low until the next update event, UEV, occurs.

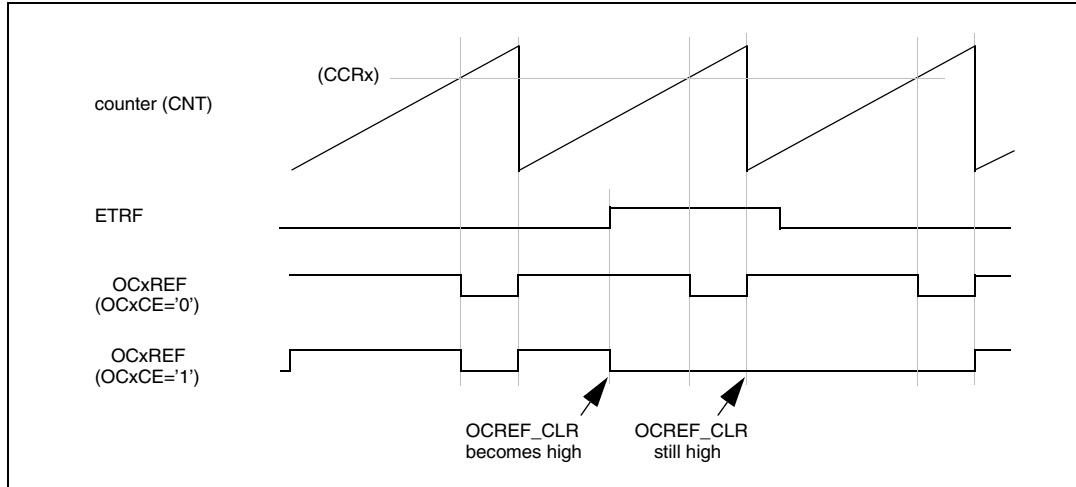
This function can only be used in output compare and PWM modes, and does not work in forced mode.

For example, the OC_xREF signal) can be connected to the output of a comparator to be used for current handling. In this case, the ETR must be configured as follow:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIM_x_SMCR register set to '00'.
2. The external clock mode 2 must be disabled: bit ECE of the TIM_x_SMCR register set to '0'.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

Figure 103 shows the behavior of the OC_xREF signal when the ETRF Input becomes High, for both values of the enable bit OC_xCE. In this example, the timer TIM_x is programmed in PWM mode.

Figure 103. Clearing TIM_x OC_xREF



Note: *In case of a PWM with a 100% duty cycle (if CCR_x>ARR), then OC_xREF is enabled again at the next counter overflow.*

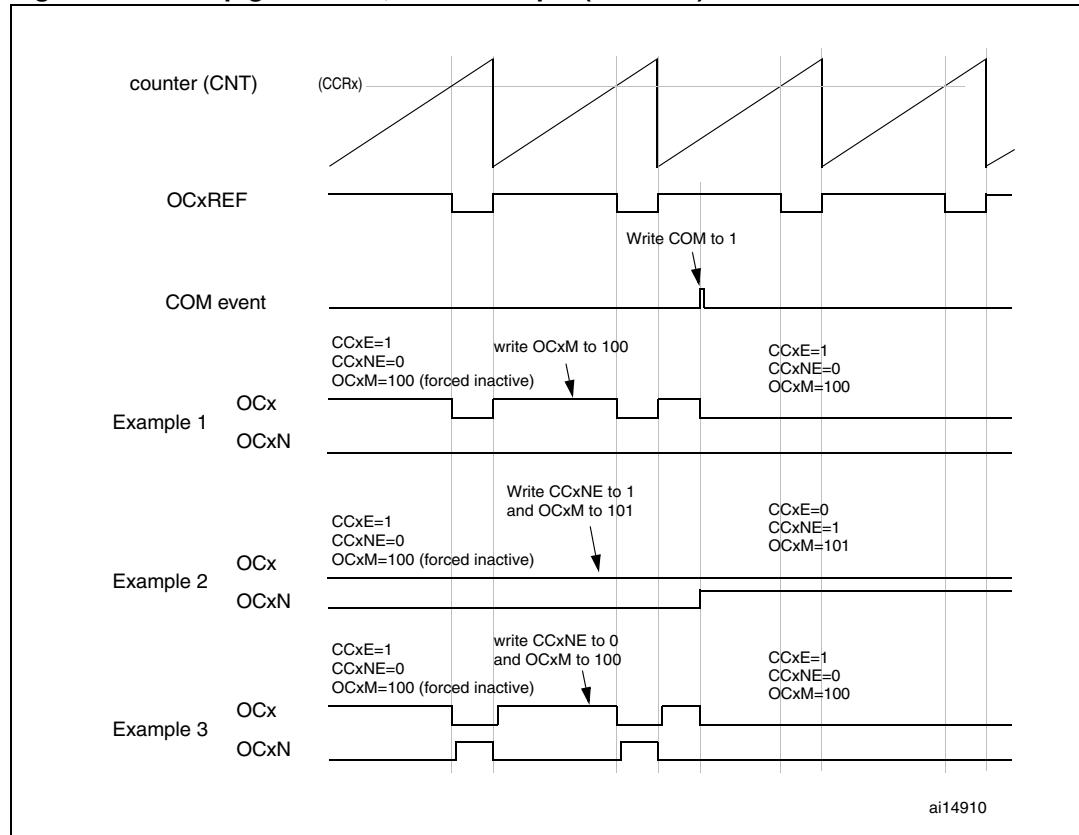
13.3.14 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus you can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

Figure 104 describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 104. 6-step generation, COM example (OSSR=1)



13.3.15 One-pulse mode

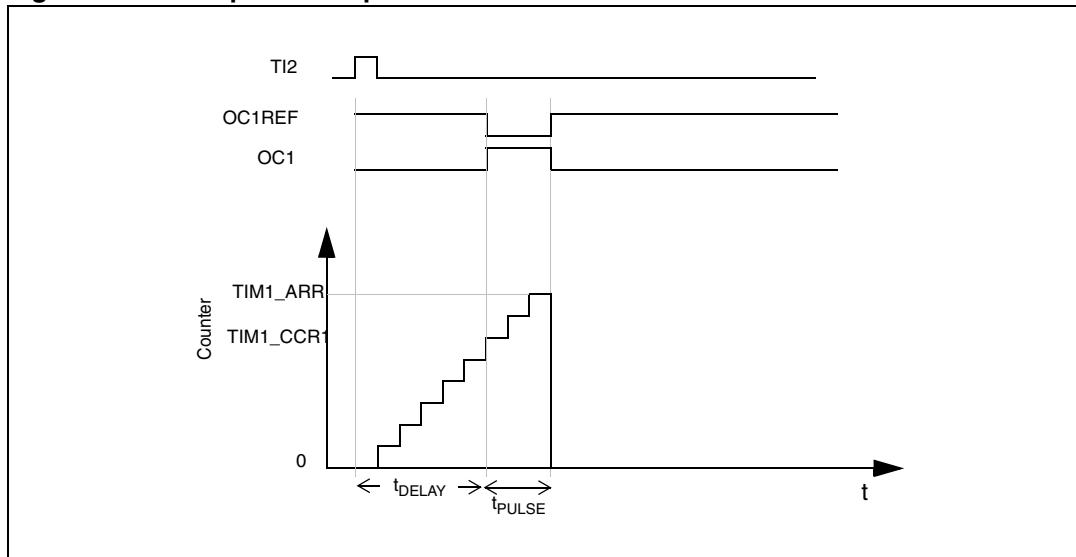
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx \leq ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 105. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P='0' 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) 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.

13.3.16 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, you can program the input filter as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to [Table 55](#). The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So you must

configure TIMx_ARR before starting. in the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 don't switch at the same time.

Table 55. 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 106 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).

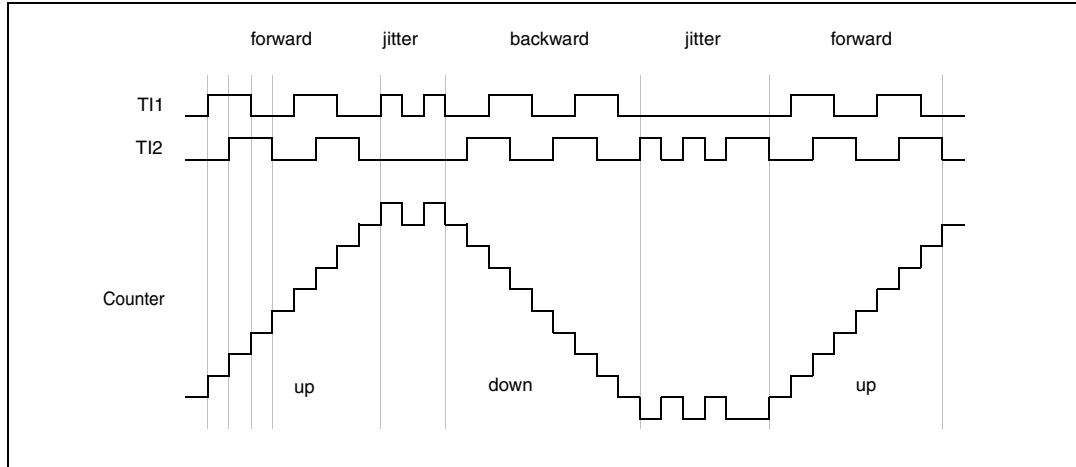
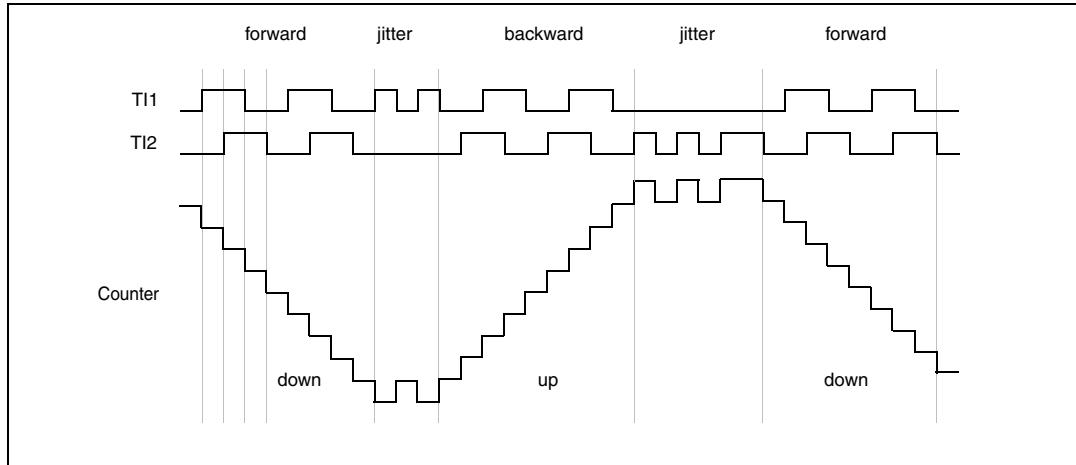
Figure 106. Example of counter operation in encoder interface mode.

Figure 107 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 107. Example of encoder interface mode with TI1FP1 polarity inverted.

The timer, when configured in Encoder Interface mode provides information on the sensor's current position. You can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. You can do this by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request generated by a real-time clock.

13.3.17 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. An example of this feature used to interface Hall sensors is given in [Section 13.3.18](#) below.

13.3.18 Interfacing with Hall sensors

This is done using the advanced-control timers (TIM1 or TIM8) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3, TIM4 or TIM5) referred to as “interfacing timer” in [Figure 108](#). 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 91: Capture/compare channel \(example: channel 1 input stage\) on page 309](#)). 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 or TIM8) (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 or TIM8) 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.

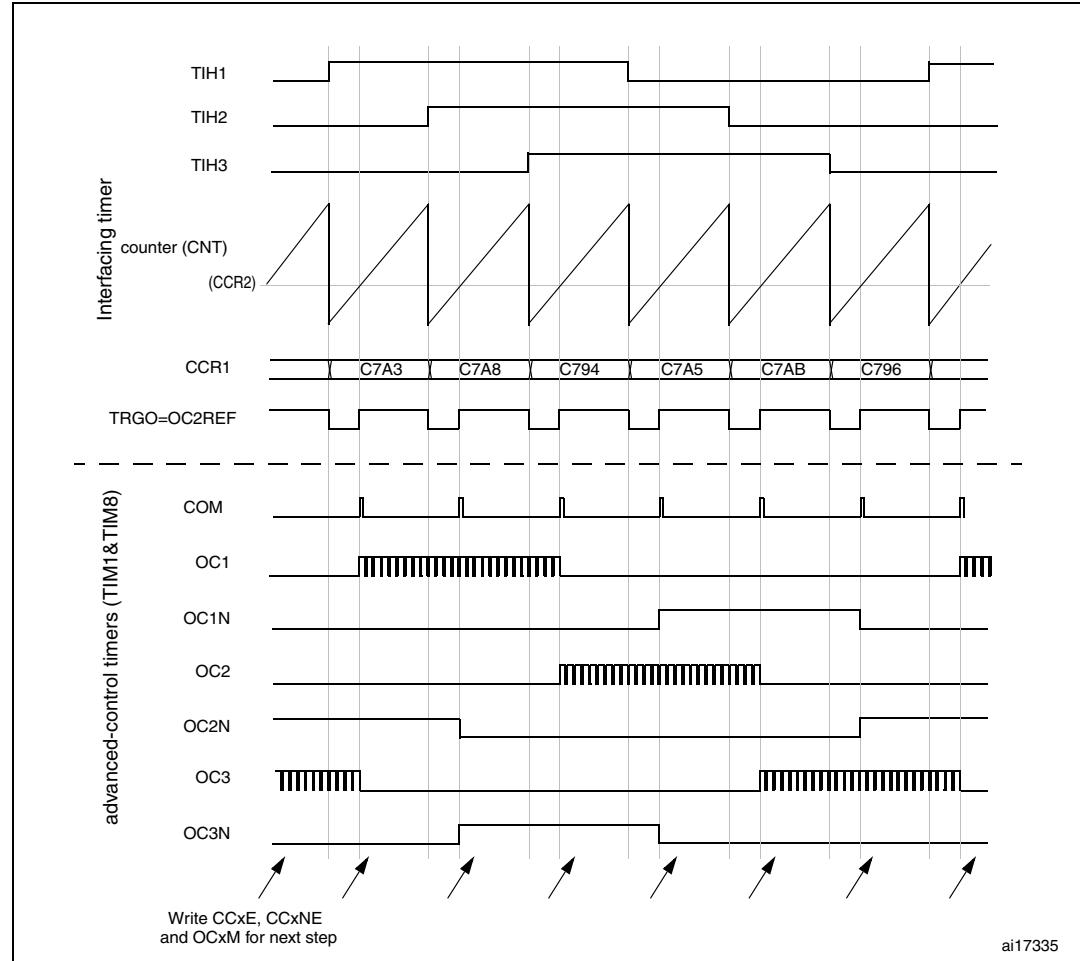
- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 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).

Figure 108 describes this example.

Figure 108. Example of hall sensor interface



13.3.19 TIMx and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

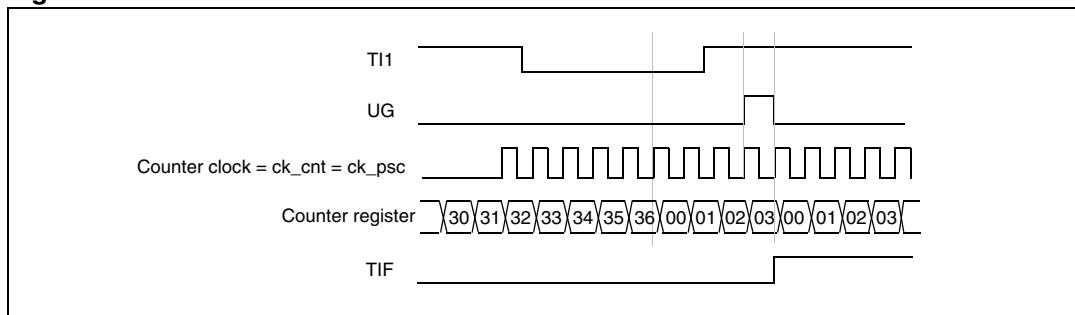
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 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 109. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

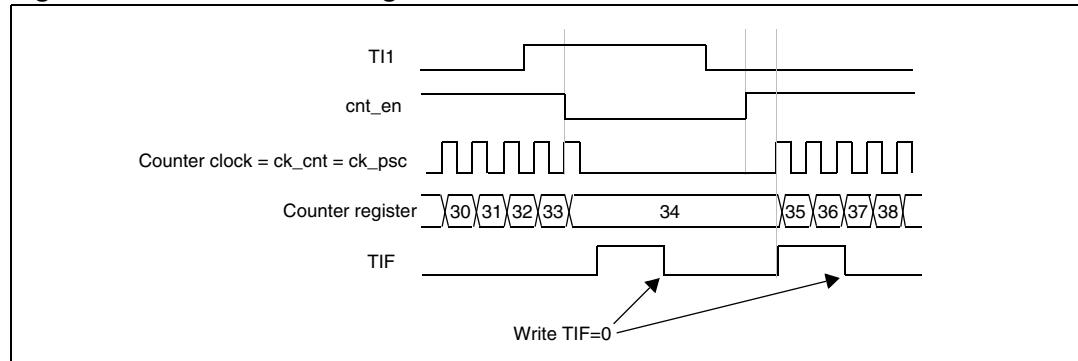
In the following example, the upcounter counts only when TI1 input is low:

- 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 110. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

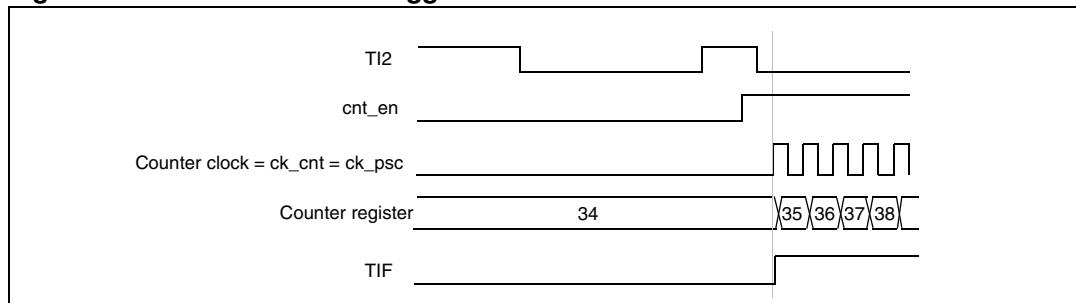
In the following example, the upcounter starts in response to a rising edge on TI2 input:

- 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 111. Control circuit in trigger mode



Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

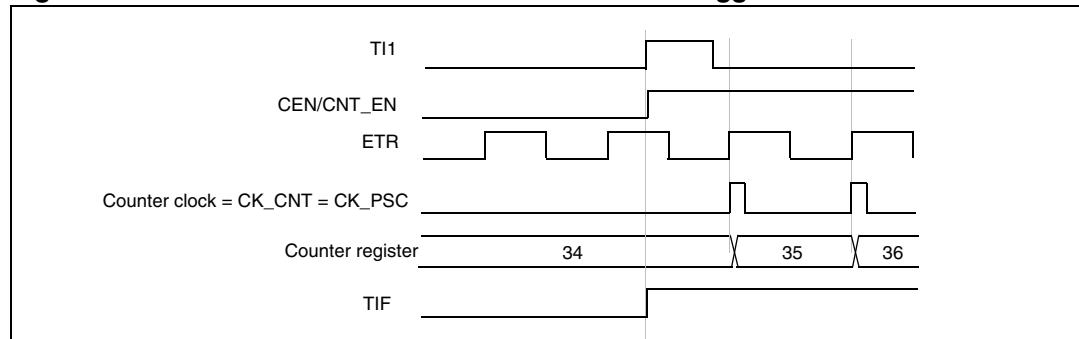
- Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
 - ETF = 0000: no filter
 - ETPS=00: prescaler disabled
 - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.

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 112. Control circuit in external clock mode 2 + trigger mode



13.3.20 Timer synchronization

The TIM timers are linked together internally for timer synchronization or chaining. Refer to [Section 14.3.15: Timer synchronization on page 391](#) for details.

13.3.21 Debug mode

When the microcontroller enters debug mode (Cortex™-M4F 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 32.16.2: Debug support for timers, watchdog, bxCAN and I²C](#).

13.4 TIM1&TIM8 registers

Refer to [Section 1.1 on page 46](#) for a list of abbreviations used in register descriptions.

13.4.1 TIM1&TIM8 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CKD[1:0]	ARPE	CMS[1:0]	DIR	OPM	URS	UDIS	CEN			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:8 CKD[1:0]: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (ETR, TIx),

00: $t_{DTS}=t_{CK_INT}$

01: $t_{DTS}=2*t_{CK_INT}$

10: $t_{DTS}=4*t_{CK_INT}$

11: Reserved, do not program this value

Bit 7 ARPE: Auto-reload preload enable

0: TIMx_ARR register is not buffered

1: TIMx_ARR register is buffered

Bits 6:5 CMS[1:0]: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 DIR: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 OPM: One pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 URS: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled.

These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 UDIS: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 CEN: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

13.4.2 TIM1&TIM8 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S		MMS[2:0]		CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OIS4:** Output Idle state 4 (OC4 output)

refer to OIS1 bit

Bit 13 **OIS3N:** Output Idle state 3 (OC3N output)

refer to OIS1N bit

Bit 12 **OIS3:** Output Idle state 3 (OC3 output)

refer to OIS1 bit

Bit 11 **OIS2N:** Output Idle state 2 (OC2N output)

refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)

refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input

1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[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, 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.

13.4.3 TIM1&TIM8 slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]			MSM	TS[2:0]			Res.	SMS[2:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Res.	rw	rw	rw

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or \overline{ETR} is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge.

1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

Note: 1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).

2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).

3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at f_{DTS}
- 0001: $f_{SAMPLING} = f_{CK_INT}$, N=2
- 0010: $f_{SAMPLING} = f_{CK_INT}$, N=4
- 0011: $f_{SAMPLING} = f_{CK_INT}$, N=8
- 0100: $f_{SAMPLING} = f_{DTS}/2$, N=6
- 0101: $f_{SAMPLING} = f_{DTS}/2$, N=8
- 0110: $f_{SAMPLING} = f_{DTS}/4$, N=6
- 0111: $f_{SAMPLING} = f_{DTS}/4$, N=8
- 1000: $f_{SAMPLING} = f_{DTS}/8$, N=6
- 1001: $f_{SAMPLING} = f_{DTS}/8$, N=8
- 1010: $f_{SAMPLING} = f_{DTS}/16$, N=5
- 1011: $f_{SAMPLING} = f_{DTS}/16$, N=6
- 1100: $f_{SAMPLING} = f_{DTS}/16$, N=8
- 1101: $f_{SAMPLING} = f_{DTS}/32$, N=5
- 1110: $f_{SAMPLING} = f_{DTS}/32$, N=6
- 1111: $f_{SAMPLING} = f_{DTS}/32$, N=8

Bit 7 **MSM**: Master/slave mode

- 0: No action
- 1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

- 000: Internal Trigger 0 (ITR0)
- 001: Internal Trigger 1 (ITR1)
- 010: Internal Trigger 2 (ITR2)
- 011: Internal Trigger 3 (ITR3)
- 100: TI1 Edge Detector (TI1F_ED)
- 101: Filtered Timer Input 1 (TI1FP1)
- 110: Filtered Timer Input 2 (TI2FP2)
- 111: External Trigger input (ETRF)

See [Table 56: TIMx Internal trigger connection on page 339](#) for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMS:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.

001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Table 56. TIMx Internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM5	TIM2	TIM3	TIM4
TIM8	TIM1	TIM2	TIM4	TIM5

13.4.4 TIM1&TIM8 DMA/interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE:** Trigger DMA request enable

0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE:** COM DMA request enable

0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE:** Capture/Compare 4 DMA request enable

0: CC4 DMA request disabled
1: CC4 DMA request enabled

- Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled
- Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled
- Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled
- Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled
- Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled
- Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled
- Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled
- Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled
- Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled
- Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled
- Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled
- Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

13.4.5 TIM1&TIM8 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CC4OF	CC3OF	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
				rc_w0	rc_w0	rc_w0	rc_w0	Res.	rc_w0							

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred.
1: An active level has been detected on the break input.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is 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 13.4.3: TIM1&TIM8 slave mode control register \(TIMx_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx_CR1 register.

13.4.6 TIM1&TIM8 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 TG: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 COMG: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels having a complementary output.

Bit 4 CC4G: Capture/Compare 4 generation

refer to CC1G description

Bit 3 CC3G: Capture/Compare 3 generation

refer to CC1G description

Bit 2 CC2G: Capture/Compare 2 generation

refer to CC1G description

Bit 1 CC1G: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 UG: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

13.4.7 TIM1&TIM8 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]	OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]			
IC2F[3:0]			IC2PSC[1:0]				IC1F[3:0]			IC1PSC[1:0]						
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	

Output compare mode:

Bit 15 **OC2CE**: Output Compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE**: Output Compare 1 clear enable

OC1CE: Output Compare 1 Clear Enable

0: OC1Ref is not affected by the ETRF Input

1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M:** Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

100: Force inactive level - OC1REF is forced low.

101: Force active level - OC1REF is forced high.

110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF='1').

111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

2: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

3: 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. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 CC1S: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

Bits 15:12 IC2F: Input capture 2 filter

Bits 11:10 IC2PSC[1:0]: Input capture 2 prescaler

Bits 9:8 CC2S: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 IC1F[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).

13.4.8 TIM1&TIM8 capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]	OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]				
IC4F[3:0]			IC4PSC[1:0]				IC3F[3:0]			IC3PSC[1:0]							
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw		rw	rw		

Output compare mode

Bit 15 **OC4CE:** Output compare 4 clear enable

Bits 14:12 **OC4M:** Output compare 4 mode

Bit 11 **OC4PE:** Output compare 4 preload enable

Bit 10 **OC4FE:** Output compare 4 fast enable

Bits 9:8 **CC4S:** Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 6:4 **OC3M:** Output compare 3 mode

Bit 3 **OC3PE:** Output compare 3 preload enable

Bit 2 **OC3FE:** Output compare 3 fast enable

Bits 1:0 **CC3S:** Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

Input capture mode

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

13.4.9 TIM1&TIM8 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output polarity
refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable
refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity
refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable
refer to CC1NE description

Bit 9 **CC3P**: Capture/Compare 3 output polarity
refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable
refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
refer to CC1NP description

Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable
refer to CC1NE description

Bit 5 **CC2P**: Capture/Compare 2 output polarity
refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

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: 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.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.

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 TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

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 or encoder 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). This configuration must not be used in encoder mode.

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.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **CC1E**: Capture/Compare 1 output enable

CC1 channel configured as output:

0: Off - OC1 is not active. OC1 level is then function of MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSS1, 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 57. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states ⁽¹⁾	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	0	0	0	Output Disabled (not driven by the timer) OCx=0, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	0	1	Output Disabled (not driven by the timer) OCx=0, OCx_EN=0	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1
		1	0	0	Output Disabled (not driven by the timer) OCx=CCxP, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=CCxNP, OCxN_EN=0
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP, OCx_EN=1	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1
0	X	0	0	0	Output Disabled (not driven by the timer) OCx=CCxP, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=CCxNP, OCxN_EN=0
		0	0	1	Output Disabled (not driven by the timer)	
		0	1	0	Asynchronously: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCxN_EN=0	
		0	1	1	Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state.	
		1	0	0	Output Disabled (not driven by the timer) OCx=CCxP, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=CCxNP, OCxN_EN=0
		1	0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1	
		1	1	0	Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1	
		1	1	1	Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	

- When both outputs of a channel are not used ($CCxE = CCxNE = 0$), the $OISx$, $OISxN$, $CCxP$ and $CCxNP$ bits must be kept cleared.

Note: *The state of the external I/O pins connected to the complementary OCx and $OCxN$ channels depends on the OCx and $OCxN$ channel state and the GPIO registers.*

13.4.10 TIM1&TIM8 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]:** Counter value

13.4.11 TIM1&TIM8 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”).

13.4.12 TIM1&TIM8 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 [Section 13.3.1: Time-base unit on page 295](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

13.4.13 TIM1&TIM8 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000

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

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

Bits 7:0 REP[7:0]: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

- the number of PWM periods in edge-aligned mode
- the number of half PWM period in center-aligned mode.

13.4.14 TIM1&TIM8 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 CCR1[15:0]: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

13.4.15 TIM1&TIM8 capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

13.4.16 TIM1&TIM8 capture/compare register 3 (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (IC3).

13.4.17 TIM1&TIM8 capture/compare register 4 (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (IC4).

13.4.18 TIM1&TIM8 break and dead-time register (TIMx_BDTR)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTG[7:0]															
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		rw							

Note: As the bits AOE, BKP, BKE, OSSR, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: OC and OCN outputs are disabled or forced to idle state.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).

See OC/OCN enable description for more details ([Section 13.4.9: TIM1&TIM8 capture/compare enable register \(TIMx_CCER\) on page 348](#)).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

- 0: Break input BRK is active low
- 1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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 13.4.9: TIM1&TIM8 capture/compare enable register \(TIMx_CCER\) on page 348](#)).

- 0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0).
- 1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1. Then, OC/OCN enable output signal=1

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details ([Section 13.4.9: TIM1&TIM8 capture/compare enable register \(TIMx_CCER\) on page 348](#)).

- 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}=2x t_{DTS}.

DTG[7:5]=110 => DT=(32+DTG[4:0])x t_{dtg} with T_{dtg}=8x t_{DTS}.

DTG[7:5]=111 => DT=(32+DTG[4:0])x t_{dtg} with T_{dtg}=16x t_{DTS}.

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

13.4.19 TIM1&TIM8 DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DBL[4:0]					Reserved	DBA[4:0]					rw	rw	rw	rw
	rw	rw	rw	rw	rw		rw	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-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

13.4.20 TIM1&TIM8 DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address
 $(\text{TIMx_CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4$

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

Example of how to use the DMA burst feature

In this example the timer DMA burst feature is used to update the contents of the CCRx registers ($x = 2, 3, 4$) with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
 - DMA channel peripheral address is the DMAR register address
 - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
 - Number of data to transfer = 3 (See note below).
 - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:
 $\text{DBL} = 3$ transfers, $\text{DBA} = 0xE$.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

Note: This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

13.4.21 TIM1&TIM8 register map

TIM1&TIM8 registers are mapped as 16-bit addressable registers as described in the table below:

Table 58. TIM1&TIM8 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x00	TIMx_CR1	Reserved												CKD [1:0]	ARPE	CMS [1:0]	DIR	OPM	URS	UDIS	CCP1	CCUS	Reserved	0	0	0	0	0	0	0	0					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x04	TIMx_CR2	Reserved												OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	T1S	MMS[2:0]	CCDS	CCUS	Reserved	0	0	0	0	0	0					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x08	TIMx_SMCR	Reserved												ETP	ETPS [1:0]	ETF[3:0]	MSM	TS[2:0]	Reserved	SMS[2:0]	0	0	0	0	0	0	0	0	0	0						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x0C	TIMx_DIER	Reserved												TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	0	0	0	0	0	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					
0x10	TIMx_SR	Reserved												CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	BIE	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	0	0	0	0	0	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					
0x14	TIMx_EGR	Reserved												TG	TIF	OC1CE	OC1M [2:0]	OC1PE	OC1FE	BG	CC4G	CC3G	CC2G	CC1G	CC1IF	0	0	0	0	0	0					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x18	TIMx_CCMR1 Output Compare mode	Reserved												OC2CE	OC2M [2:0]	OC2PE	OC2FE	CC2S [1:0]	OC1CE	OC1M [2:0]	OC1PE	OC1FE	BG	CC4G	CC3G	CC2G	CC1G	CC1IF	0	0	0	0	0	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					
	TIMx_CCMR1 Input Capture mode	Reserved												IC2F[3:0]	IC2 PSC [1:0]	CC2S [1:0]	IC1F[3:0]	IC1F[3:0]	IC1PSC [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					
0x1C	TIMx_CCMR2 Output Compare mode	Reserved												O24CE	OC4M [2:0]	OC4PE	OC4FE	CC4S [1:0]	OC3CE	OC3M [2:0]	OC3PE	OC3FE	OC4P	CC4G	CC3G	CC2G	CC1G	CC1IF	0	0	0	0	0	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					
	TIMx_CCMR2 Input Capture mode	Reserved												IC4F[3:0]	IC4 PSC [1:0]	CC4S [1:0]	IC3F[3:0]	IC3F[3:0]	IC3PSC [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					
0x20	TIMx_CCER	Reserved												CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3NP	CC2NP	CC2NE	CC1NP	CC1NE	CC1P	CC1E	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x24	TIMx_CNT	Reserved												CNT[15:0]												0	0	0	0	0	0	0	0			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x28	TIMx_PSC	Reserved												PSC[15:0]												0	0	0	0	0	0	0	0			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x2C	TIMx_ARR	Reserved												ARR[15:0]												0	0	0	0	0	0	0	0			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x30	TIMx_RCR	Reserved												REP[7:0]												0	0	0	0	0	0	0	0			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					

Table 58. TIM1&TIM8 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	TIMx_CCR1	Reserved												CCR1[15:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x38	TIMx_CCR2	Reserved												CCR2[15:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x3C	TIMx_CCR3	Reserved												CCR3[15:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x40	TIMx_CCR4	Reserved												CCR4[15:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x44	TIMx_BDTR	Reserved												MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]	DT[7:0]												
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x48	TIMx_DCR	Reserved												DBL[4:0]					Reserved	DBA[4:0]													
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x4C	TIMx_DMAR	Reserved												DMAB[15:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Refer to [Table 1 on page 50](#) for the register boundary addresses.

14 General-purpose timers (TIM2 to TIM5)

14.1 TIM2 to TIM5 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 14.3.15](#).

14.2 TIM2 to TIM5 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3 and TIM4) or 32-bit (TIM2 and TIM5) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
 - Input capture
 - Output compare
 - PWM generation (Edge- and Center-aligned modes)
 - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
 - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
 - Trigger event (counter start, stop, initialization or count by internal/external trigger)
 - Input capture
 - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

14.3 TIM2 to TIM5 functional description

14.3.1 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. 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.

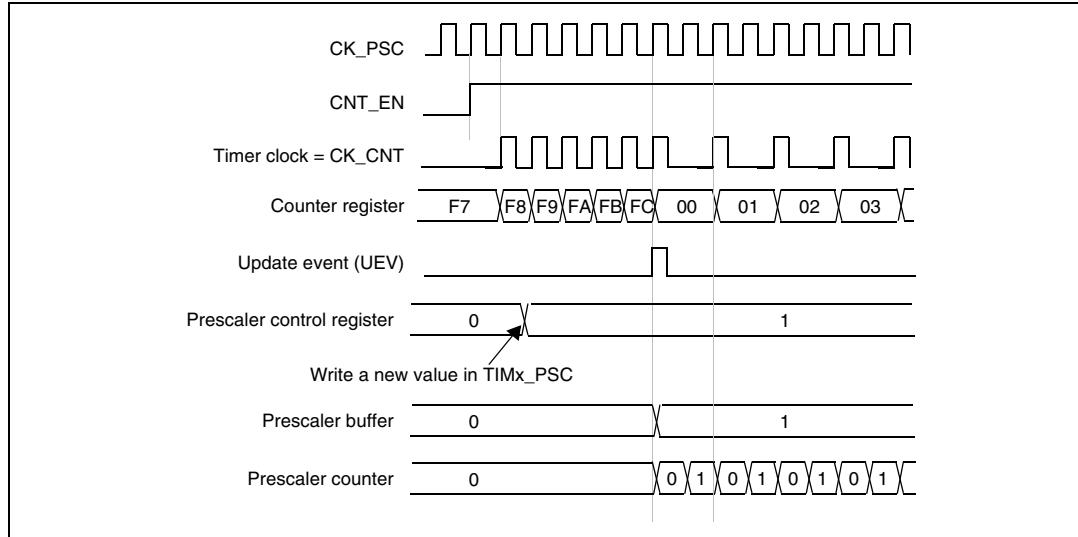
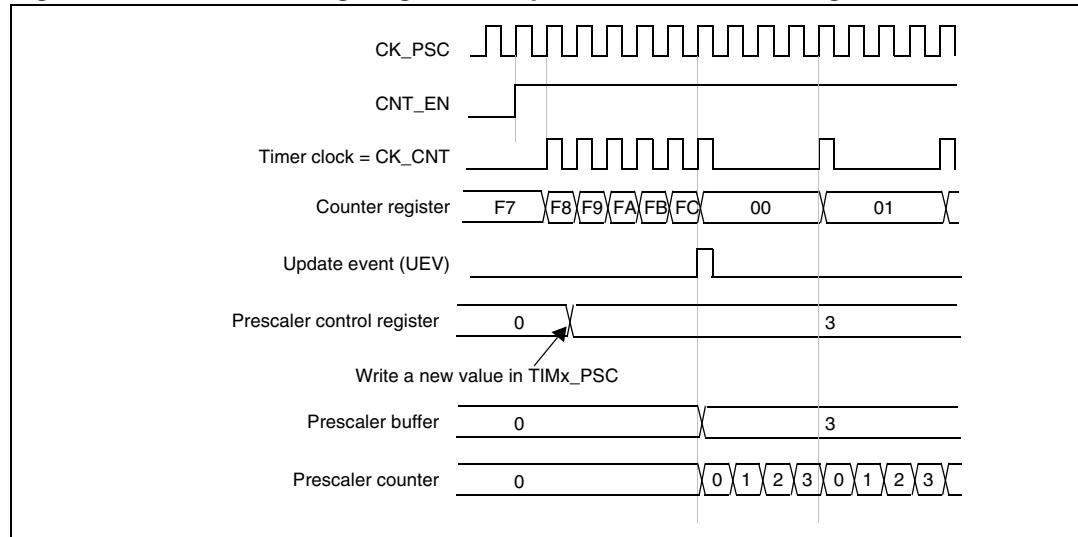
The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

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 113 and *Figure 114* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 113. Counter timing diagram with prescaler division change from 1 to 2**Figure 114. Counter timing diagram with prescaler division change from 1 to 4**

14.3.2 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 115. Counter timing diagram, internal clock divided by 1

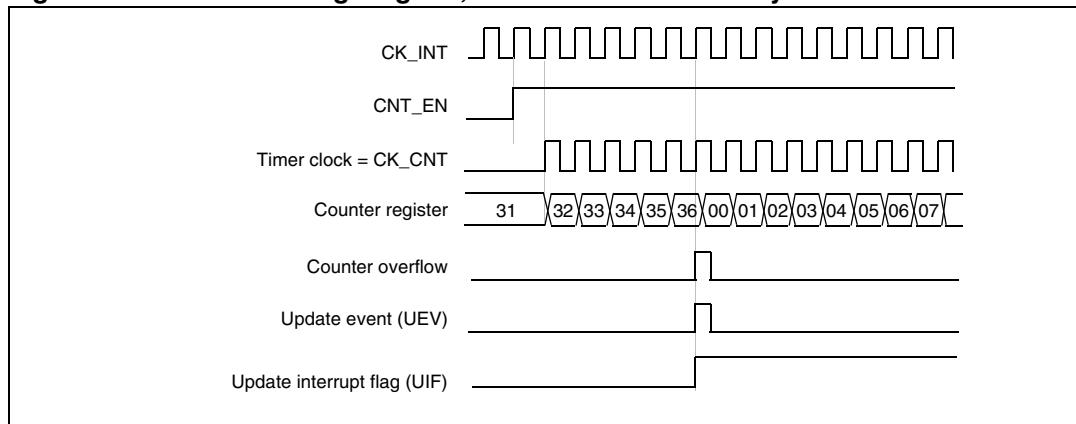


Figure 116. Counter timing diagram, internal clock divided by 2

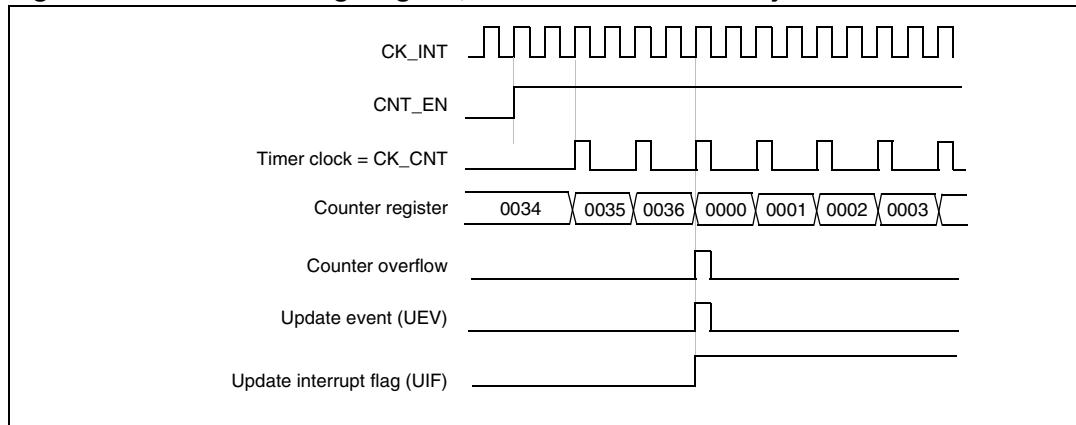


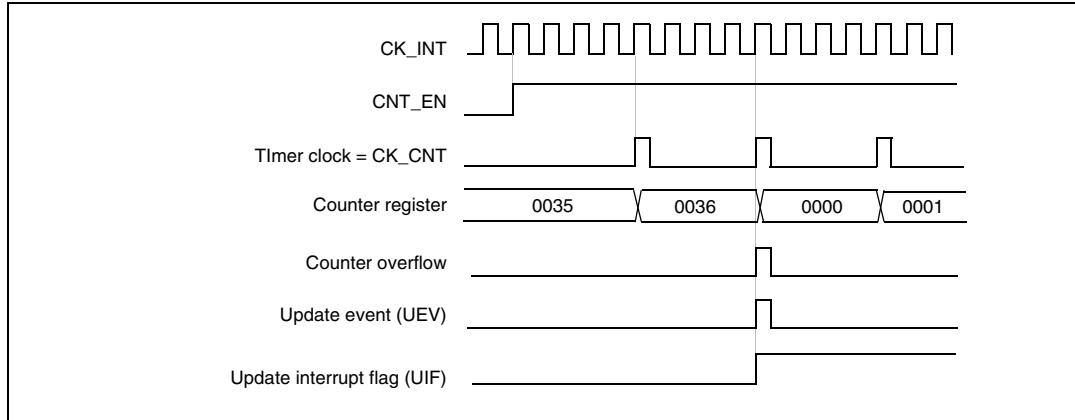
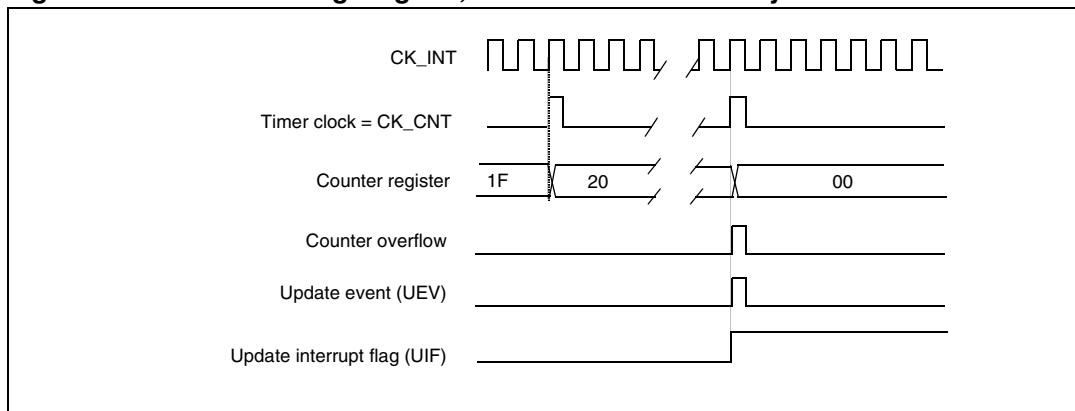
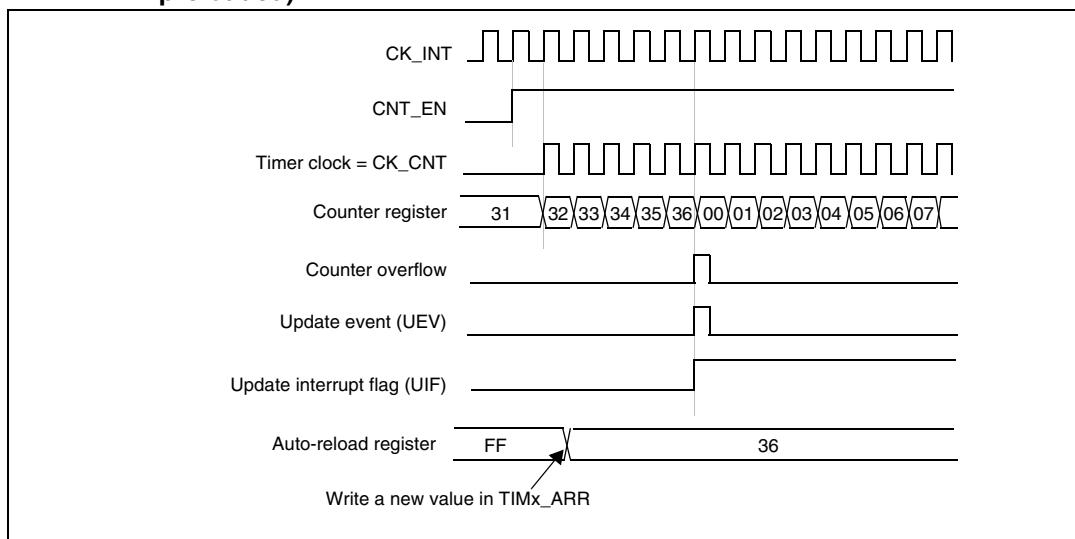
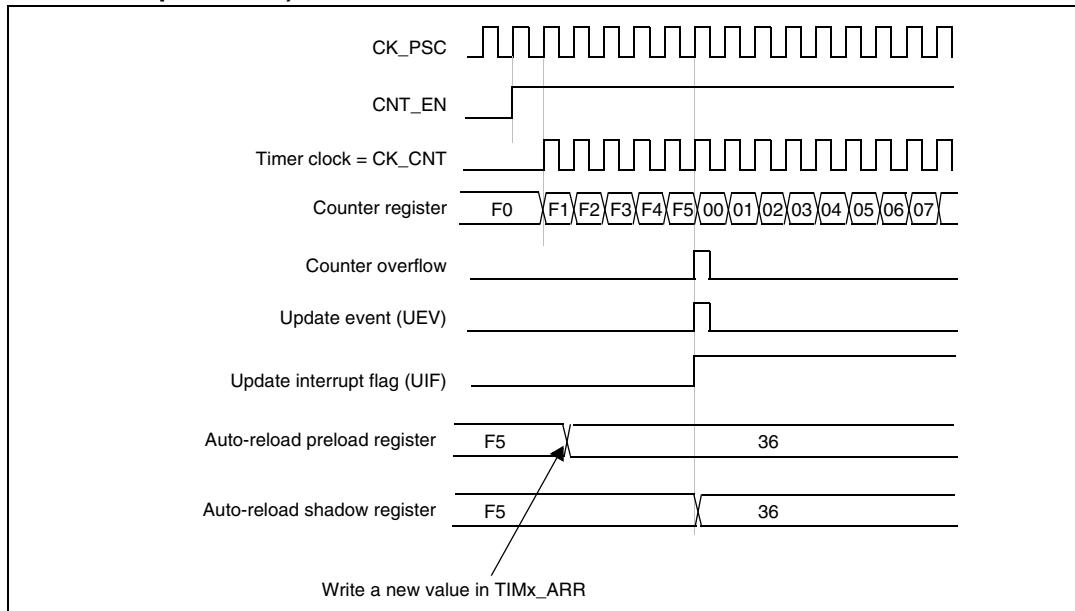
Figure 117. Counter timing diagram, internal clock divided by 4**Figure 118. Counter timing diagram, internal clock divided by N****Figure 119. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)**

Figure 120. 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.

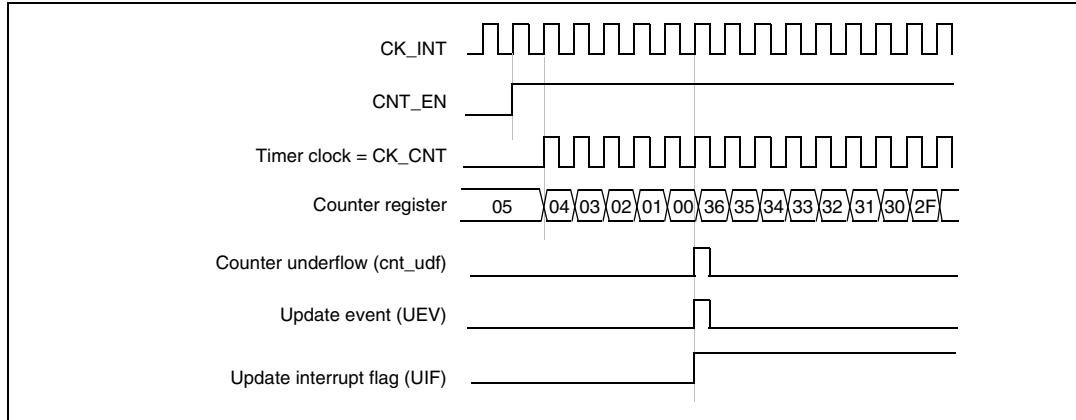
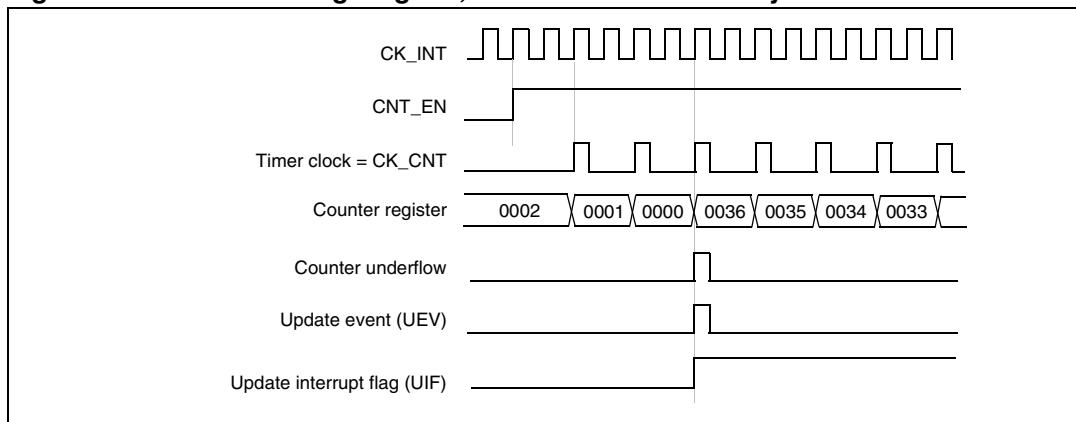
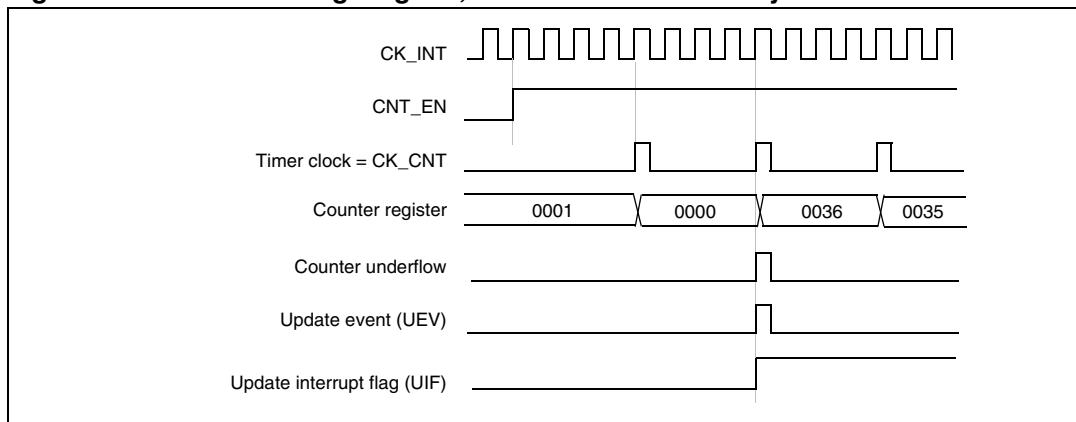
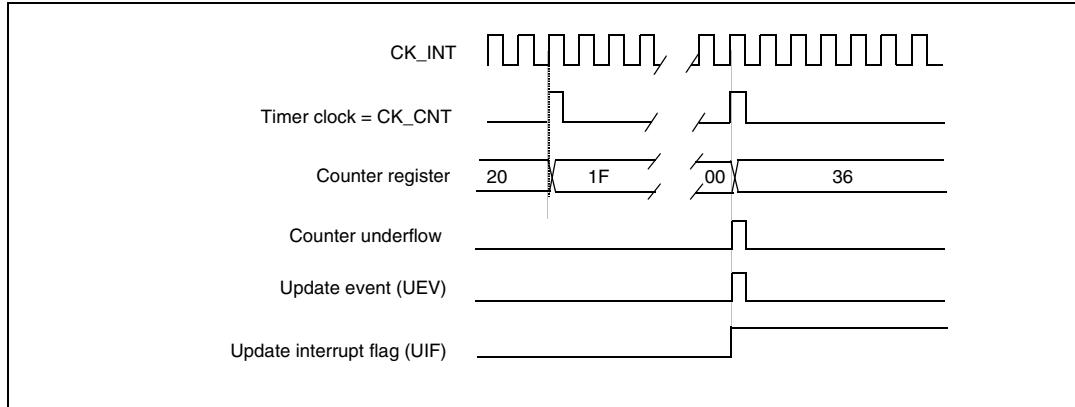
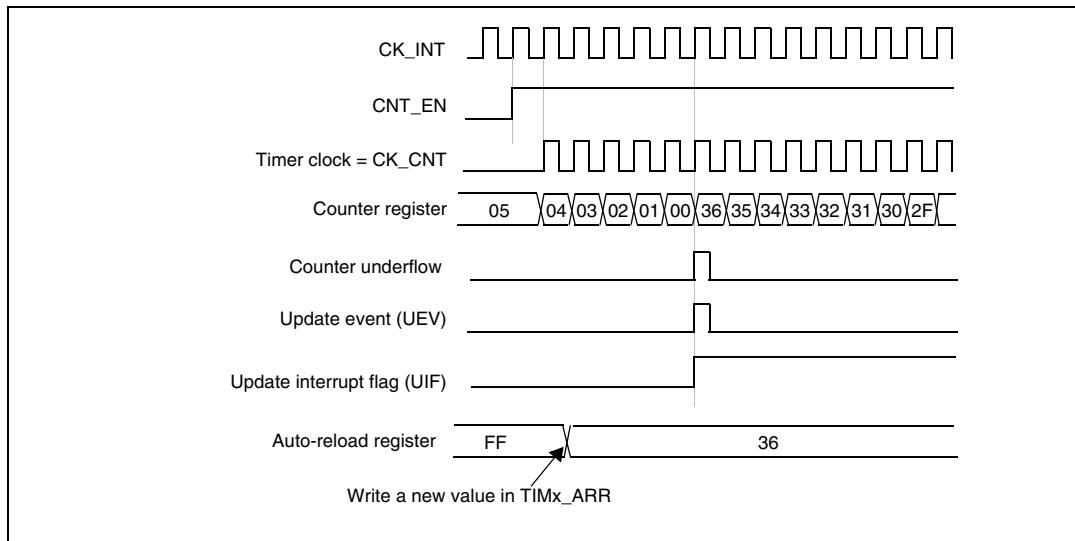
Figure 121. Counter timing diagram, internal clock divided by 1**Figure 122. Counter timing diagram, internal clock divided by 2****Figure 123. Counter timing diagram, internal clock divided by 4**

Figure 124. Counter timing diagram, internal clock divided by N**Figure 125. Counter timing diagram, Update event when repetition counter is not used**

Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") or the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the 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.

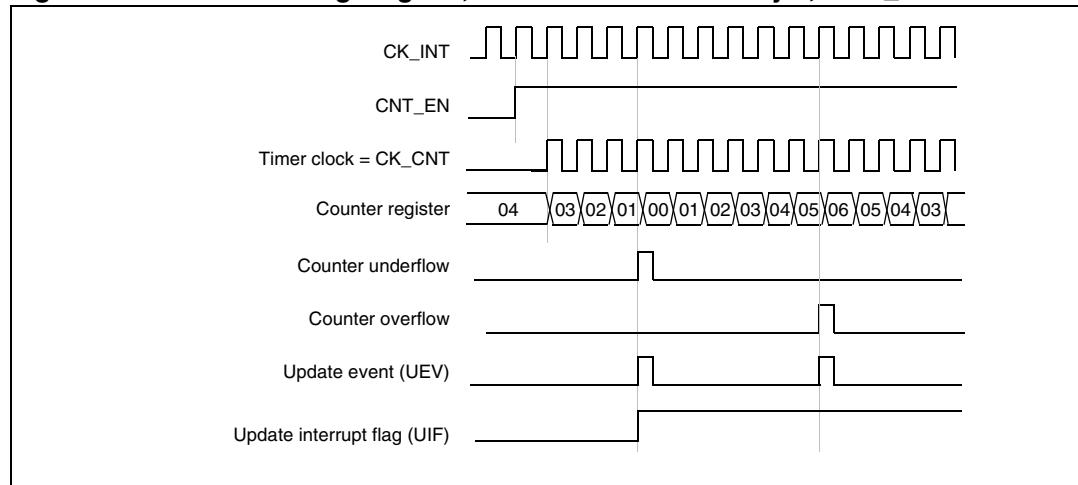
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on 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):

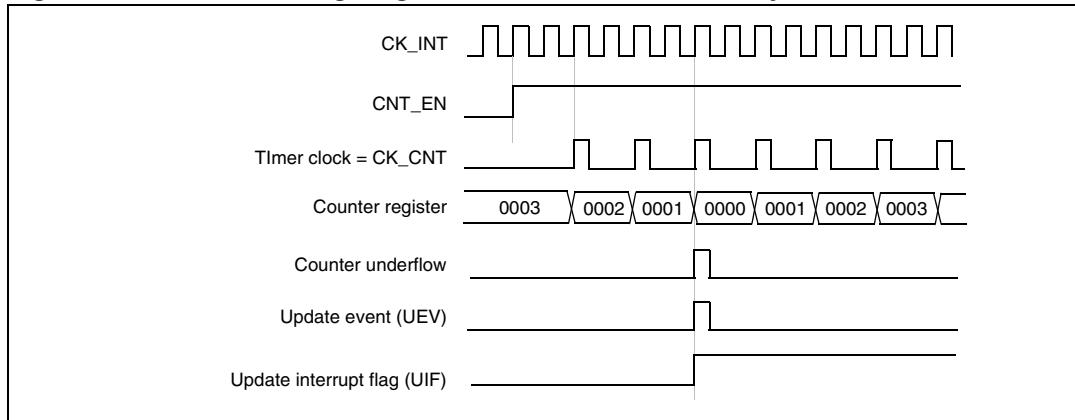
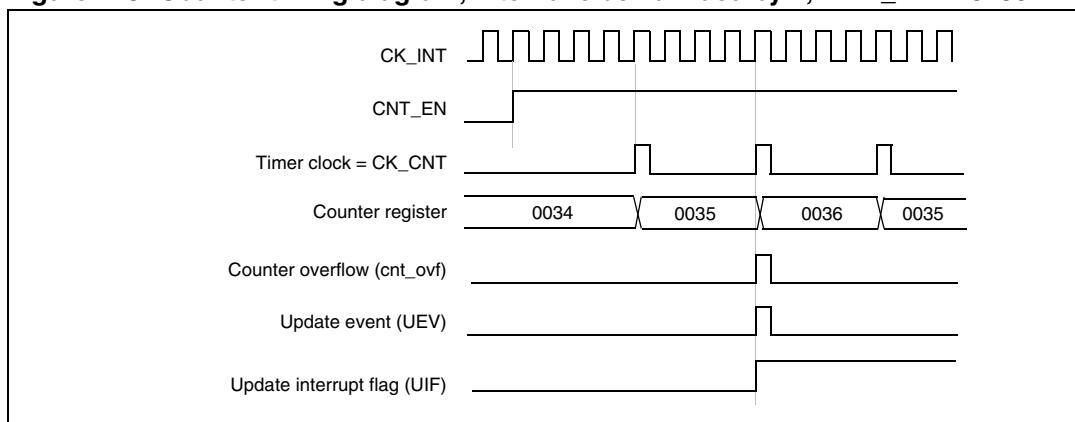
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter 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 126. 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 14.4.1: TIMx control register 1 \(TIMx_CR1\) on page 392](#)).

Figure 127. Counter timing diagram, internal clock divided by 2**Figure 128. 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.

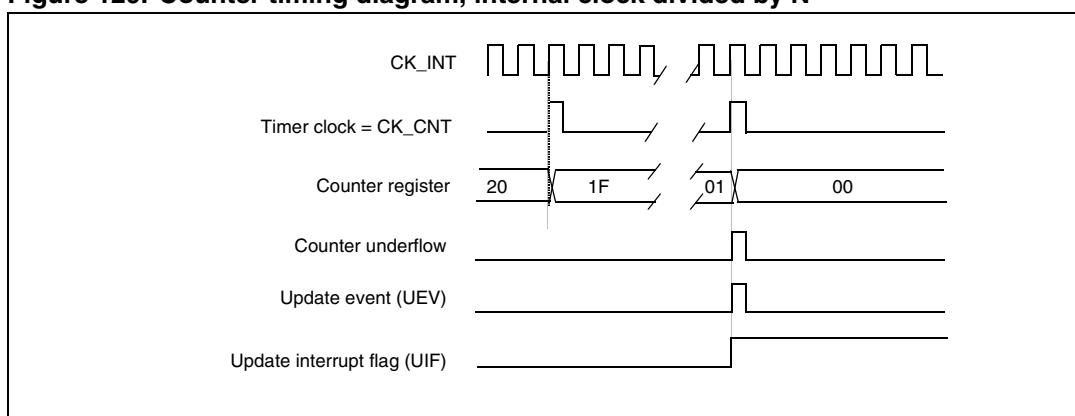
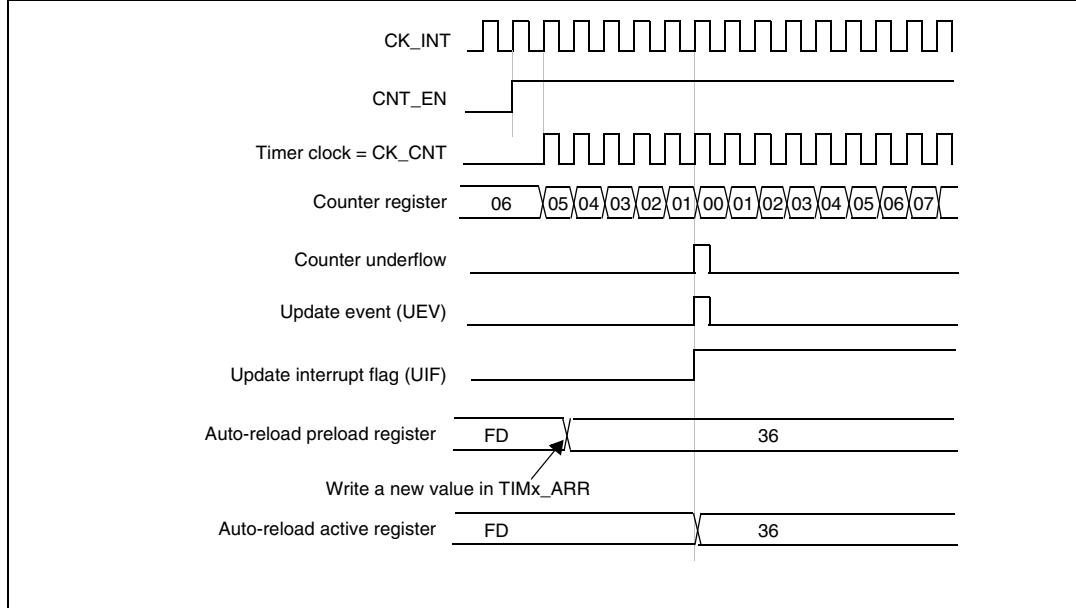
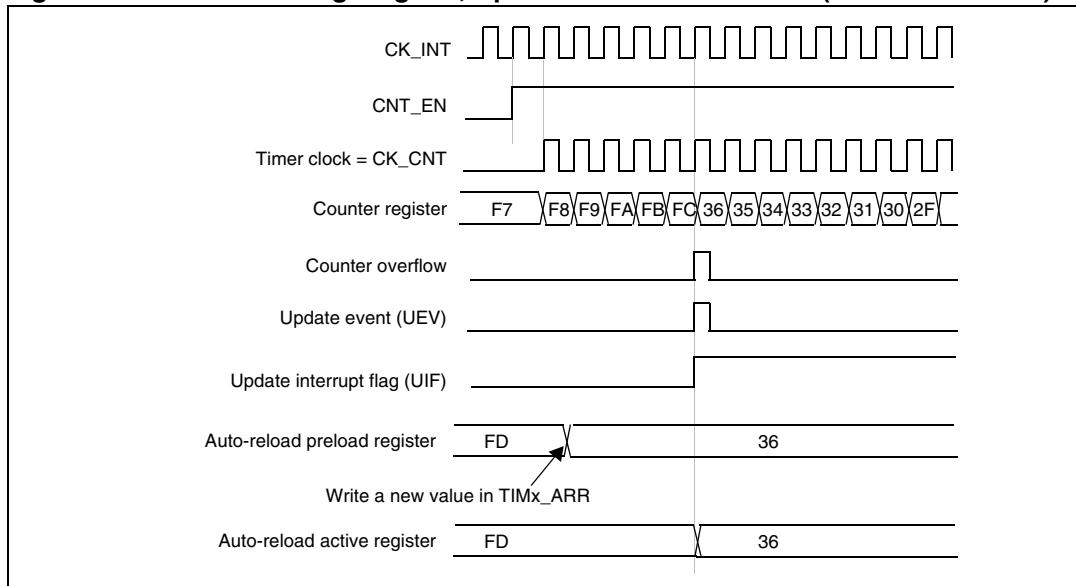
Figure 129. Counter timing diagram, internal clock divided by N

Figure 130. Counter timing diagram, Update event with ARPE=1 (counter underflow)**Figure 131. Counter timing diagram, Update event with ARPE=1 (counter overflow)**

14.3.3 Clock selection

The counter clock can be provided by the following clock sources:

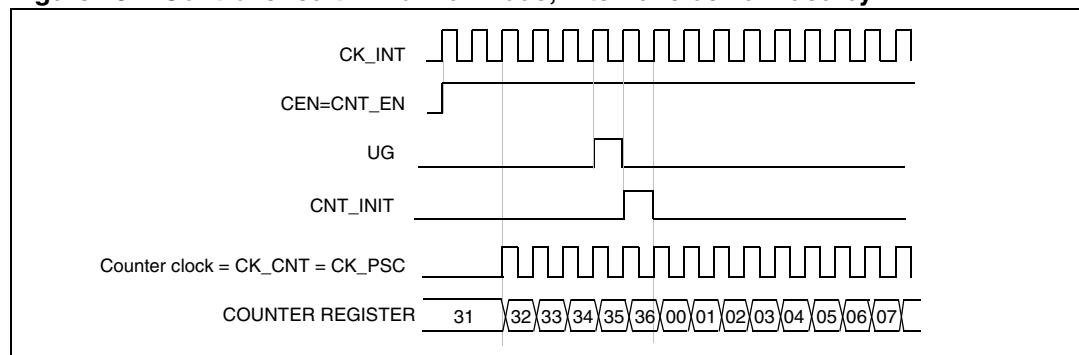
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, you can configure Timer 1 to act as a prescaler for Timer 2. Refer to : [Using one timer as prescaler for another on page 391](#) 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.

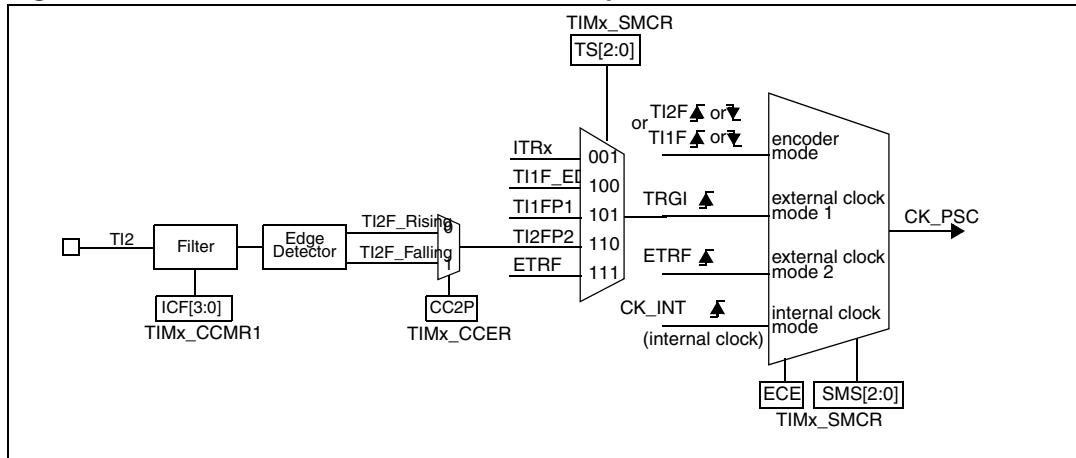
Figure 132 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 132. 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 133. TI2 external clock connection example

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= '01 in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).

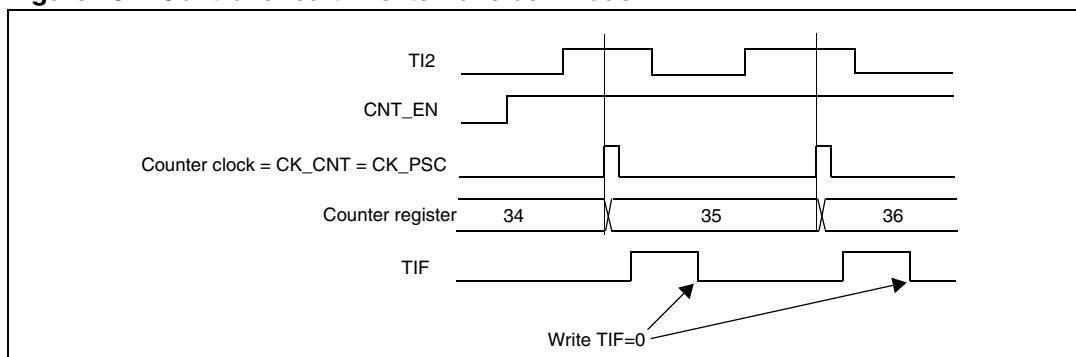
Note:

The capture prescaler is not used for triggering, so you don't need to configure it.

3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 134. Control circuit in external clock mode 1

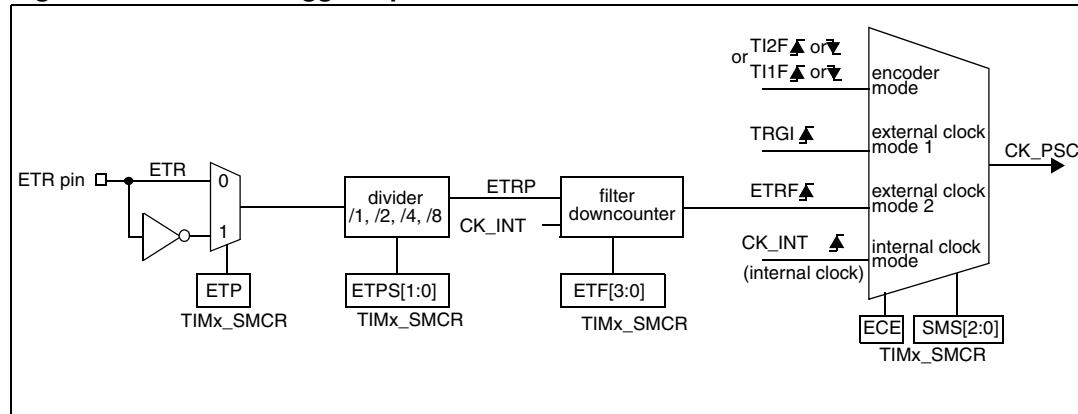
External clock source mode 2

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 135 gives an overview of the external trigger input block.

Figure 135. External trigger input block



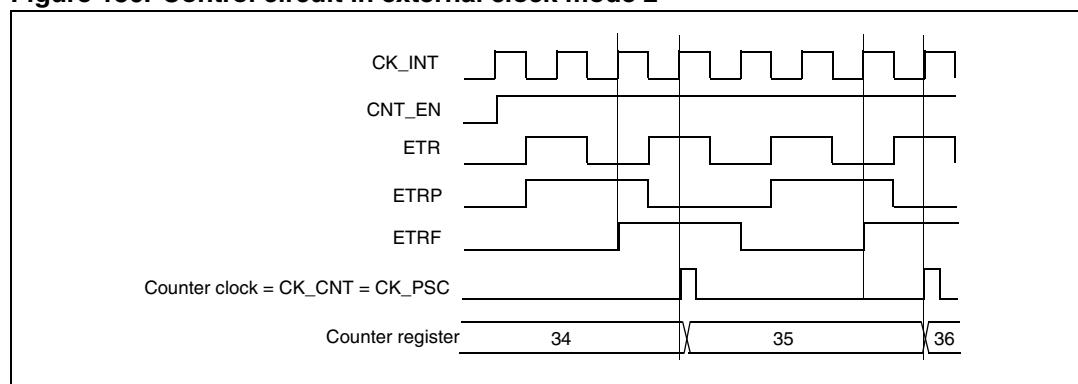
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

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 136. Control circuit in external clock mode 2



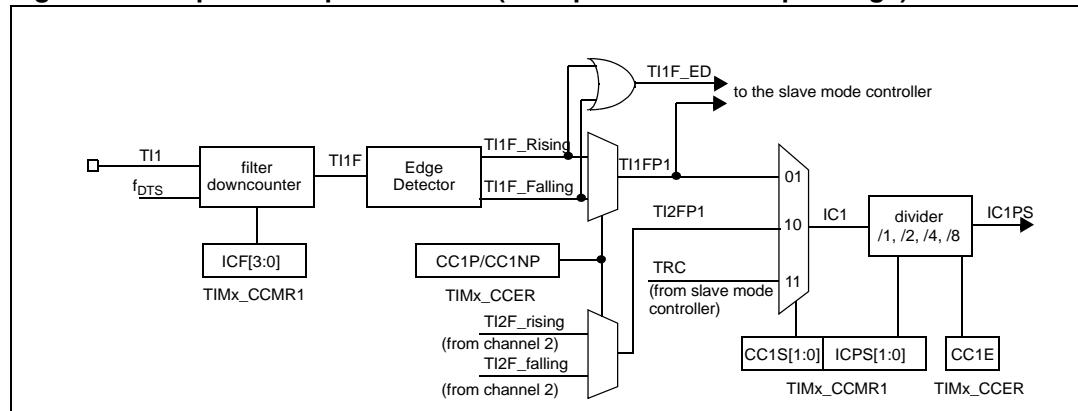
14.3.4 Capture/compare channels

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 137. 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 138. Capture/compare channel 1 main circuit

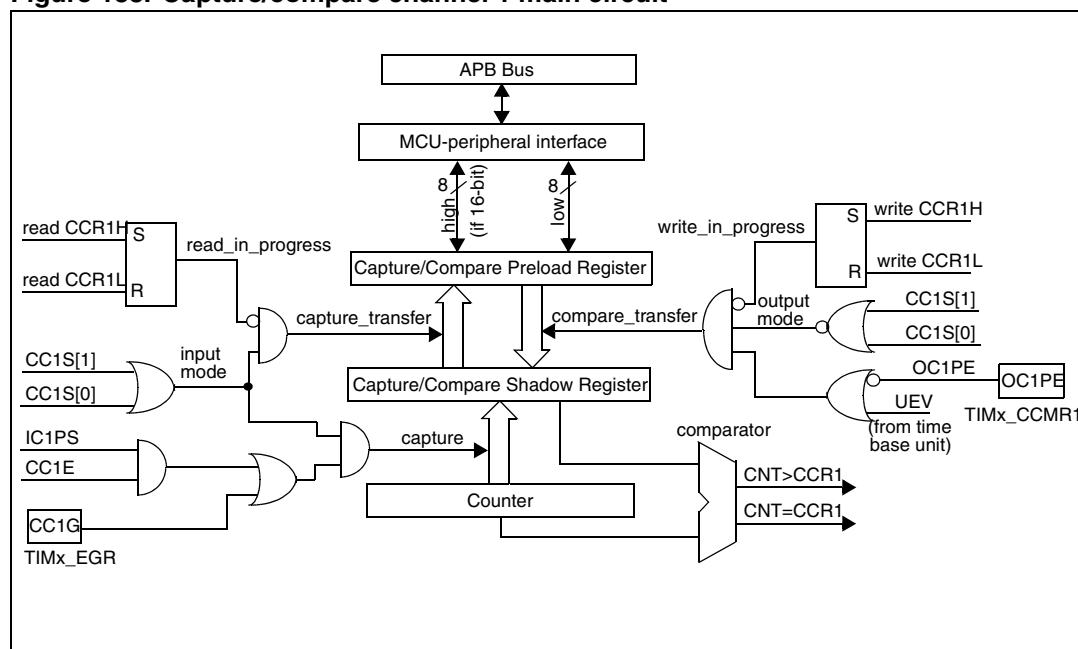
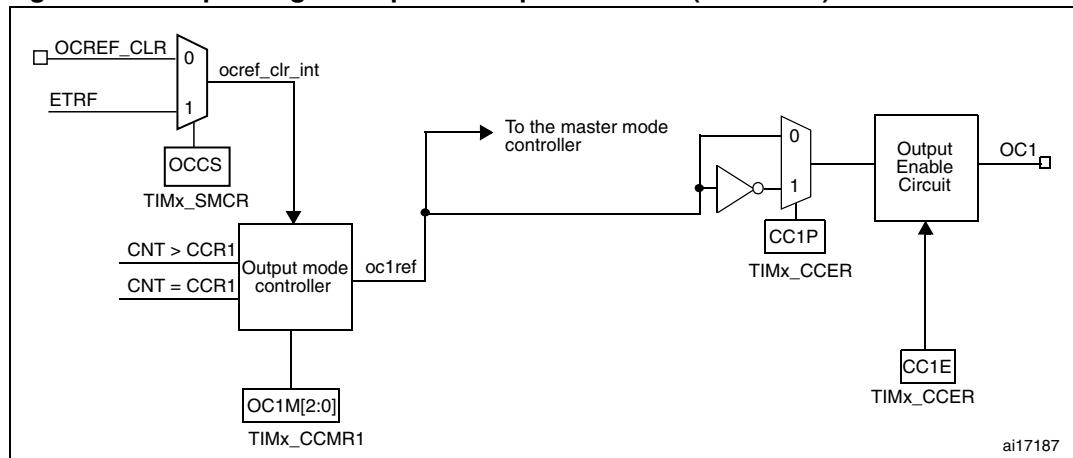


Figure 139. Output stage of capture/compare channel (channel 1)

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

14.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at 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 the CC1P and CC1NP bits to 00 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.*

14.3.6 PWM input mode

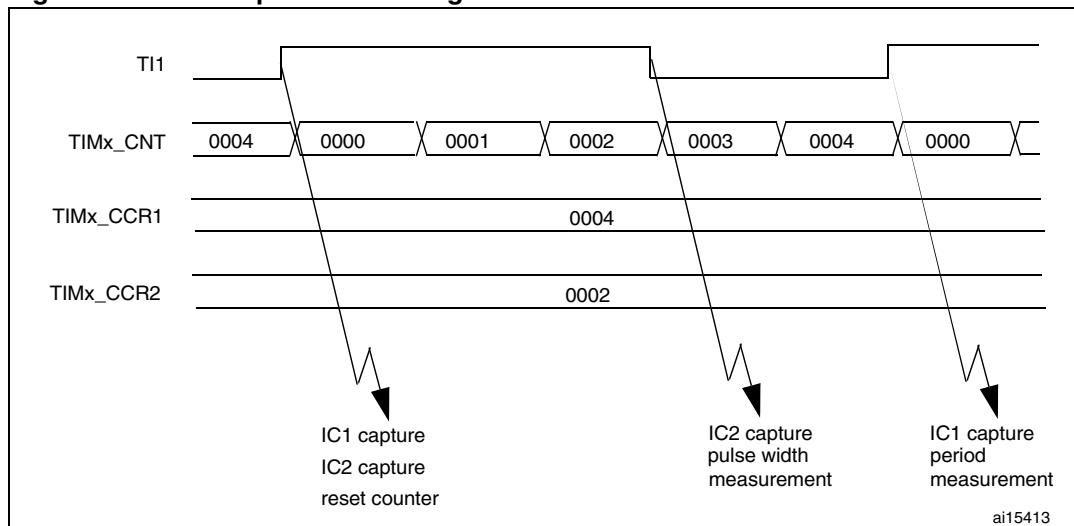
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to '0' and the CC1NP bit to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to '1' and the CC2NP bit to '0' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 140. PWM input mode timing



14.3.7 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.

14.3.8 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).

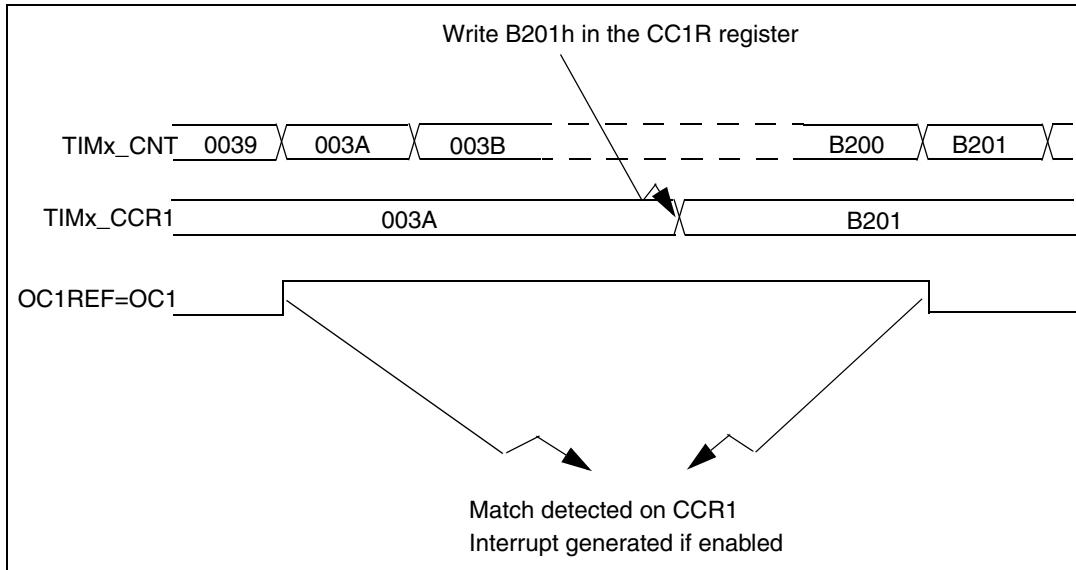
The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

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 141](#).

Figure 141. Output compare mode, toggle on OC1.

14.3.9 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_CC Rx 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_CC Rx are always compared to determine whether TIMx_CC Rx \leq TIMx_CNT or TIMx_CNT \leq TIMx_CC Rx (depending on the direction of the counter). However, to comply with the OCREF_CLR functionality (OCREF can be cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison changes, or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM='000) to one of the PWM modes (OCxM='110 or '111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

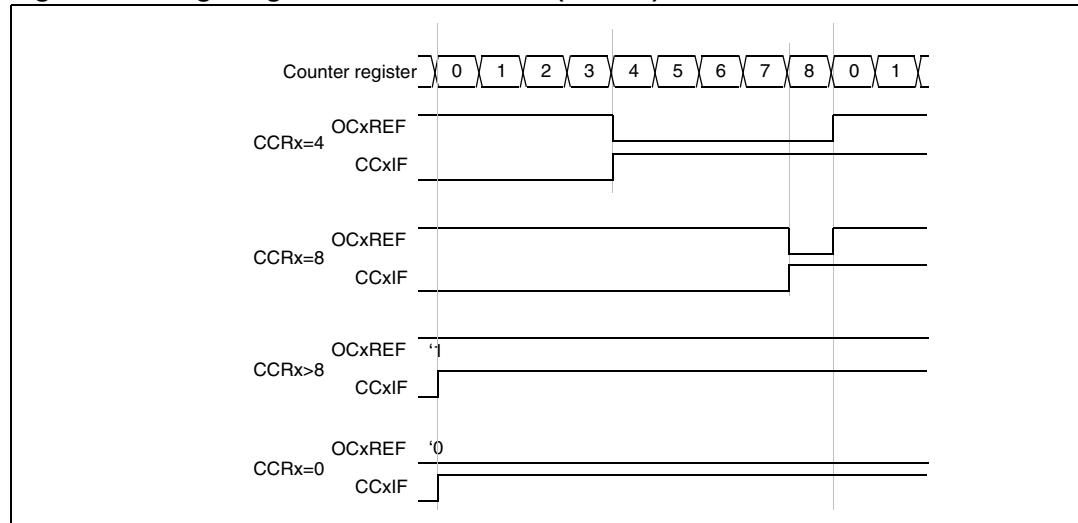
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to [Section : Upcounting mode on page 363](#).

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at '1'. If the compare value is 0 then OCxREF is held at '0'. [Figure 142](#) shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 142. Edge-aligned PWM waveforms (ARR=8)



Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to [Section : Downcounting mode on page 366](#).

In PWM mode 1, the reference signal ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then ocxref is held at '1'. 0% PWM is not possible in this mode.

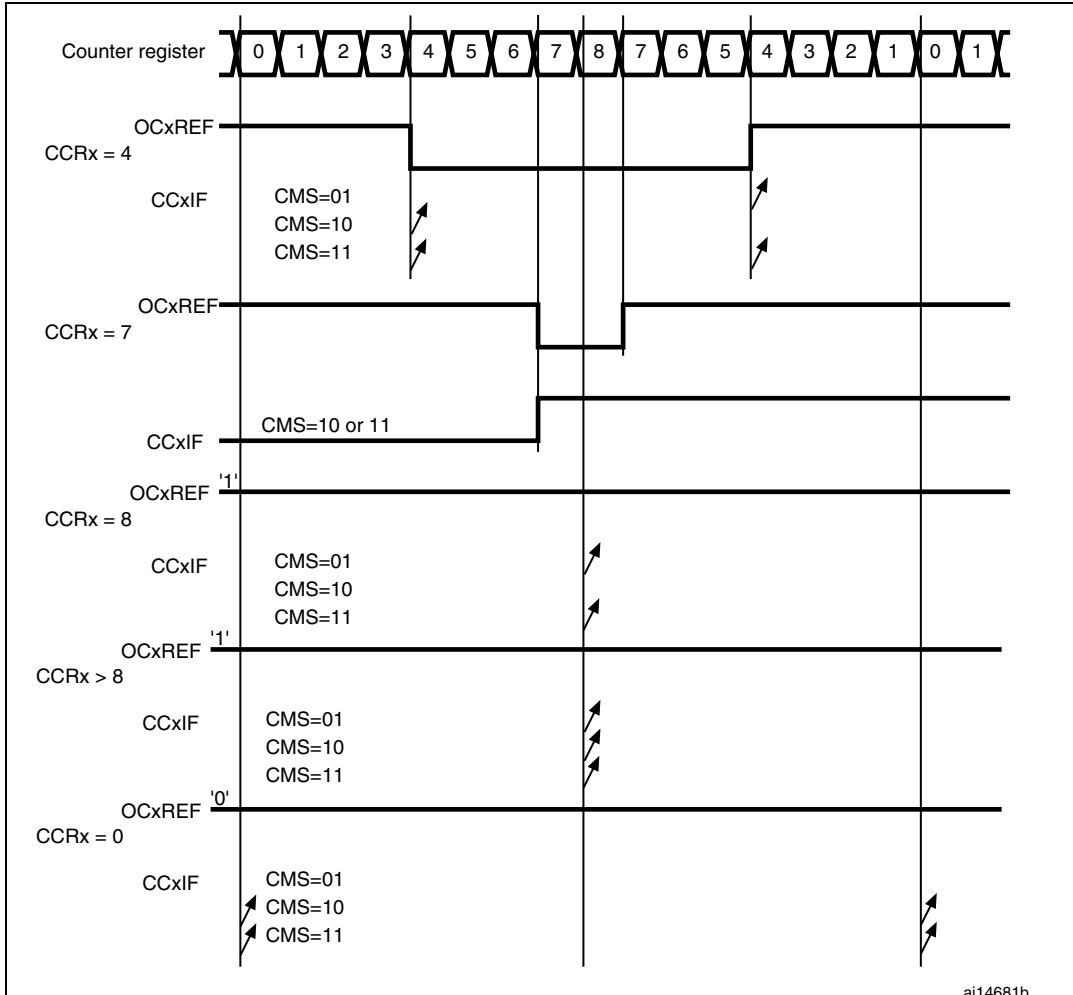
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from '00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to [Section : Center-aligned mode \(up/down counting\) on page 368](#).

[Figure 143](#) 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 143. 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.

14.3.10 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT<CCR_x≤ ARR (in particular, 0<CCR_x),
- In downcounting: CNT>CCR_x.

Figure 144. 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.

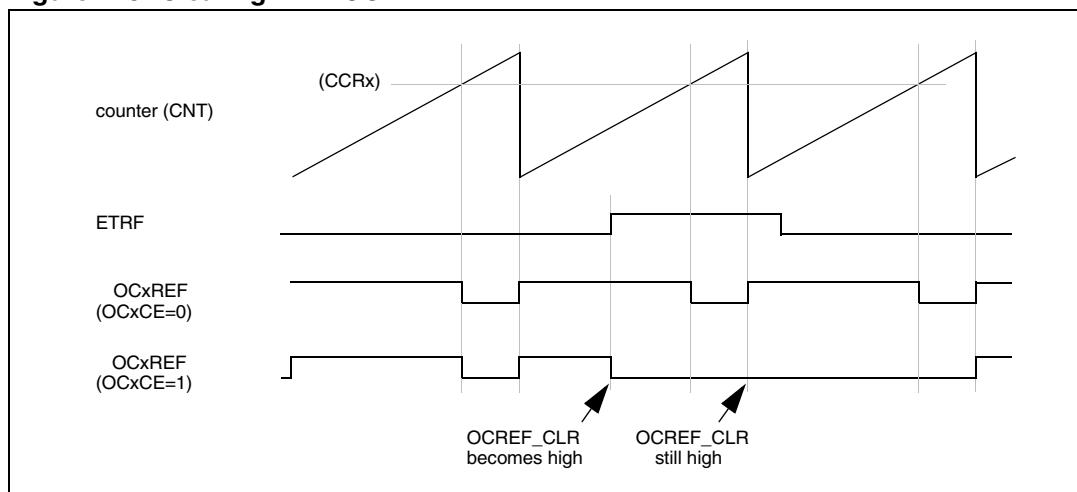
If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

14.3.11 Clearing the OCxREF signal on an external event

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the 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 145 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 145. Clearing TIMx OCxREF



1. In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.

14.3.12 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, you can program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to [Table 59](#). The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So you must configure TIMx_ARR before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 don't switch at the same time.

Table 59. 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 146](#) gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are

selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P=0, CC1NP = '0' (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P=0, CC2NP = '0' (TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)

Figure 146. Example of counter operation in encoder interface mode

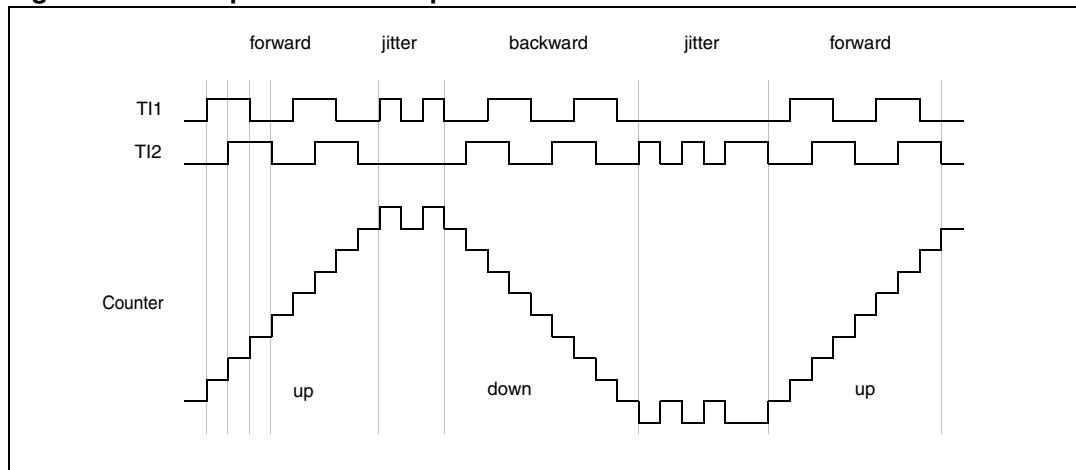
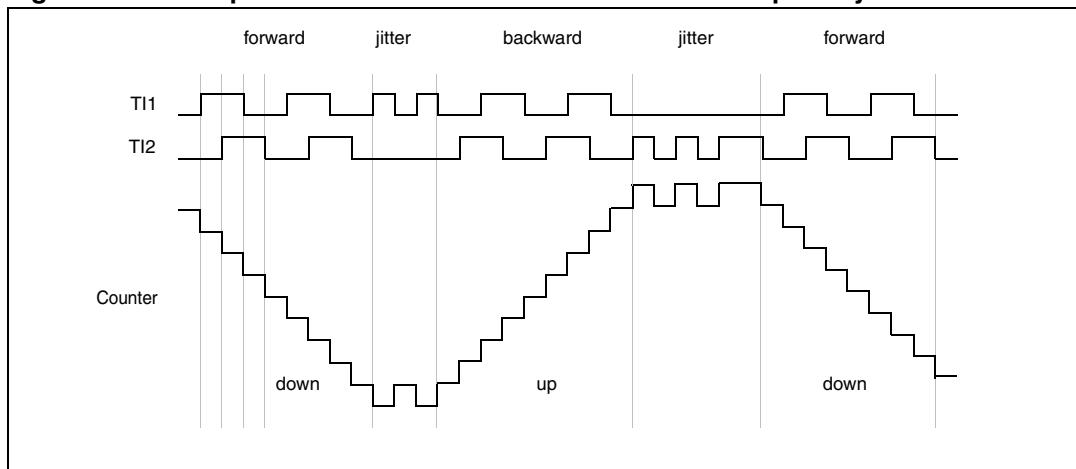


Figure 147 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 147. 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.

14.3.13 Timer input XOR function

The TI1S bit in the TIM_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.

14.3.14 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

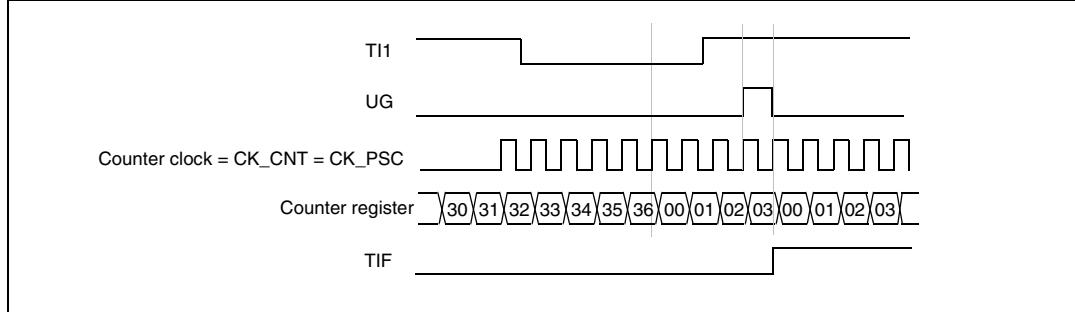
The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 148. Control circuit in reset mode**Slave mode: Gated mode**

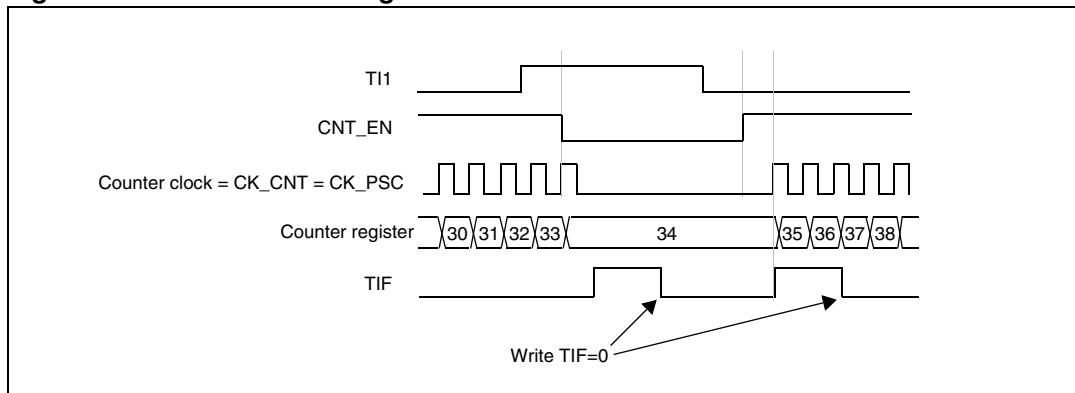
The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 149. Control circuit in gated mode

- The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

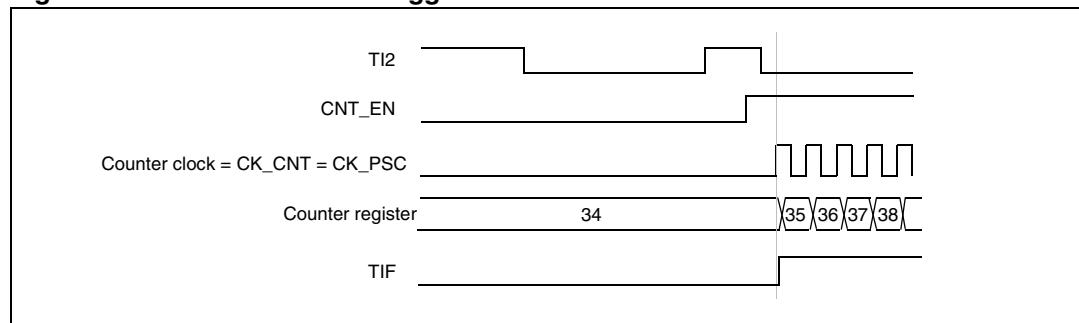
In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 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 150. Control circuit in trigger mode



Slave mode: External Clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

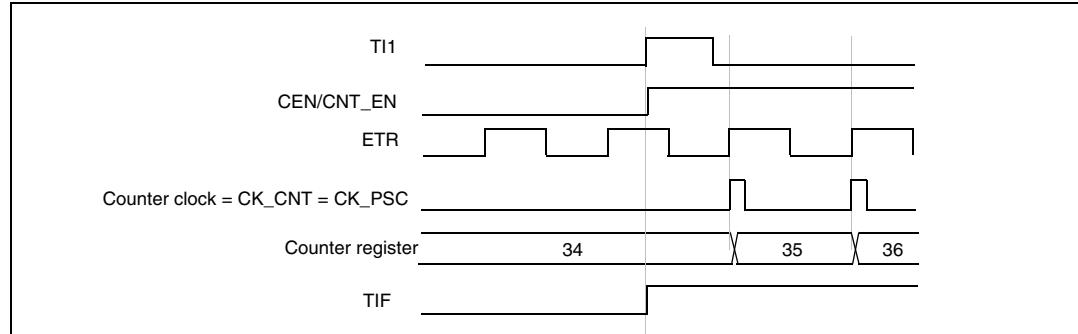
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
 - ETF = 0000: no filter
 - ETPS=00: prescaler disabled
 - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
 - IC1F=0000: no filter.
 - The capture prescaler is not used for triggering and does not need to be configured.
 - CC1S=01in TIMx_CCMR1 register to select only the input capture source
 - CC1P=0 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 151. Control circuit in external clock mode 2 + trigger mode



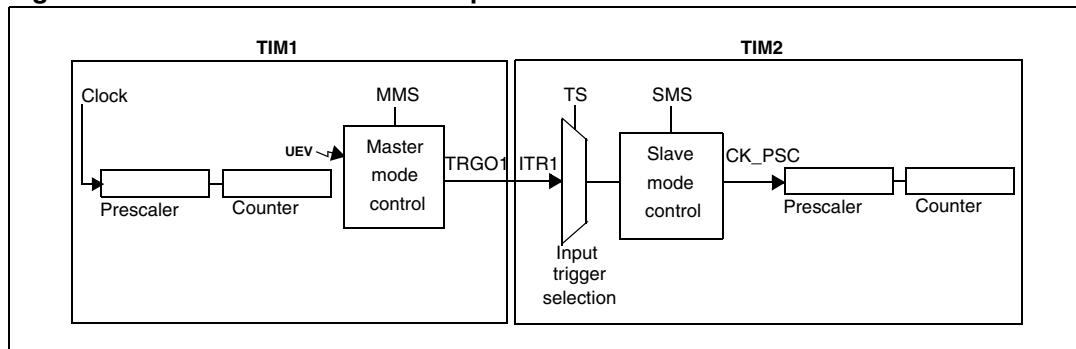
14.3.15 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 152: 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 152. Master/Slave timer example



14.3.16 Debug mode

When the microcontroller enters debug mode (Cortex™-M4F 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 32.16.2: Debug support for timers, watchdog, bxCAN and I²C](#).

14.4 TIM2 to TIM5 registers

Refer to [Section 1.1 on page 46](#) for a list of abbreviations used in register descriptions.

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

14.4.1 TIMx control register 1 (TIMx_CR1)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						Reserved	CKD[1:0]	ARPE	CMS	DIR	OPM	URS	UDIS	CEN	
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:8 CKD: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),

- 00: $t_{DTS} = t_{CK_INT}$
- 01: $t_{DTS} = 2 \times t_{CK_INT}$
- 10: $t_{DTS} = 4 \times t_{CK_INT}$
- 11: Reserved

Bit 7 ARPE: Auto-reload preload enable

- 0: TIMx_ARR register is not buffered
- 1: TIMx_ARR register is buffered

Bits 6:5 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.

14.4.2 TIMx control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS[2:0]			CCDS	Reserved		
rw								rw	rw	rw	rw	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)

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)

Bits 2:0 Reserved, must be kept at reset value.

14.4.3 TIMx slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]			MSM	TS[2:0]			Res.	SMS[2:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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:

001: Internal Trigger 1 (ITR1).

010: Internal Trigger 2 (ITR2).

011:

100: TI1 Edge Detector (TI1F_ED)

101: Filtered Timer Input 1 (TI1FP1)

110: Filtered Timer Input 2 (TI2FP2)

111: External Trigger input (ETRF)

Note: Table 60: TIMx internal trigger connection on page 395 These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bits 2:0 **SMS:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

000: Slave mode disabled - if CEN = '1 then the prescaler is clocked directly by the internal clock.

001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

14.4.4 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	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

- 0: Trigger DMA request disabled.
- 1: Trigger DMA request enabled.

Bit 13 Reserved, always read as 0

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable

- 0: CC4 DMA request disabled.
- 1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable

- 0: CC3 DMA request disabled.
- 1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable

- 0: CC2 DMA request disabled.
- 1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable

- 0: CC1 DMA request disabled.
- 1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable

- 0: Update DMA request disabled.
- 1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable

- 0: Trigger interrupt disabled.
- 1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable

- 0: CC4 interrupt disabled.
- 1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable

- 0: CC3 interrupt disabled
- 1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable

- 0: CC2 interrupt disabled
- 1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

- 0: CC1 interrupt disabled
- 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

- 0: Update interrupt disabled
- 1: Update interrupt enabled

14.4.5 TIMx status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CC4OF	CC3OF	CC2OF	CC1OF		Reserved	TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
				rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

- 0: No overcapture has been detected
- 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

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

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

- 0: No trigger event occurred
- 1: Trigger interrupt pending

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output:

This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.

0: No match

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input:

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

- This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow or underflow (for TIM2 to TIM5) and if UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

14.4.6 TIMx event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TG	Res.	CC4G	CC3G	CC2G	CC1G	UG	
								w		w	w	w	w	w	

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

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation

refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation

refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation

refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

14.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]		
	IC2F[3:0]				IC2PSC[1:0]			IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

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).

Bits 6:4 **OC1M:** Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

100: Force inactive level - OC1REF is forced low.

101: Force active level - OC1REF is forced high.

110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).

111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=00 (the channel is configured in output).

2: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC1PE:** Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: 1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=00 (the channel is configured in output).

2: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE:** Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S:** Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}	1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
0001: $f_{SAMPLING}=f_{CK_INT}$, N=2	1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
0010: $f_{SAMPLING}=f_{CK_INT}$, N=4	1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
0011: $f_{SAMPLING}=f_{CK_INT}$, N=8	1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6	1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8	1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6	1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8	1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: In current silicon revision, f_{DTS} is replaced in the formula by CK_INT when ICxF[3:0]= 1, 2 or 3.

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).

The prescaler is reset as soon as CC1E=0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

14.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]	OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]			
IC4F[3:0]			IC4PSC[1:0]				IC3F[3:0]			IC3PSC[1:0]						
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	

Output compare mode

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

Input capture mode

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

14.4.9 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 rw	Res. rw	CC4P rw	CC4E rw	CC3NP rw	Res. rw	CC3P rw	CC3E rw	CC2NP rw	Res. rw	CC2P rw	CC2E rw	CC1NP rw	Res. rw	CC1P rw	CC1E rw

Bit 15 **CC4NP**: Capture/Compare 4 output Polarity.

Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output Polarity.

refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable.

refer to CC1E description

Bit 13 **CC3NP**: Capture/Compare 3 output Polarity.

refer to CC1NP description

Bit 12 Reserved, must be kept at reset value.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: Capture/Compare 3 output Polarity.

refer to CC1P description

- Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
refer to CC1E description
- 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 61. 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 OC_x channels depends on the OC_x channel state and the GPIO registers.

14.4.10 TIMx counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]**: counter value

14.4.11 TIMx prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to $f_{CK_PSC} / (PSC[15:0] + 1)$.

PSC contains the value to be loaded in the active prescaler register at each update event.

14.4.12 TIMx auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bits 15:0 **ARR[15:0]**: value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 14.3.1: Time-base unit on page 362](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

14.4.13 TIMx capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

14.4.14 TIMx capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR 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).

14.4.15 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_CCMR 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).

14.4.16 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 and TIM5).

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_CCMR 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).

14.4.17 TIMx DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

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

14.4.18 TIMx DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address
(TIMx_CR1 address) + (DBA + DMA index) × 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

Example of how to use the DMA burst feature

In this example the timer DMA burst feature is used to update the contents of the CCRx registers ($x = 2, 3, 4$) with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
 - DMA channel peripheral address is the DMAR register address
 - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
 - Number of data to transfer = 3 (See note below).
 - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:
DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

Note:

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

14.4.19 TIM2 option register (TIM2_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ITR1_RMP		Reserved									
				rw	rw										

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

Bits 11:10 **ITR1_RMP**: Internal trigger 1 remap

Set and cleared by software.

00: TIM8_TRGOUT

01: PTP trigger output is connected to TIM2_ITR1

10: OTG FS SOF is connected to the TIM2_ITR1 input

11: OTG HS SOF is connected to the TIM2_ITR1 input

Bits 9:0 Reserved, must be kept at reset value.

14.4.20 TIM5 option register (TIM5_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved								TI4_RMP		Reserved							
rw				rw				Reserved									

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

Bits 7:6 **TI4_RMP:** Timer Input 4 remap

Set and cleared by software.

00: TIM5 Channel4 is connected to the GPIO: Refer to the Alternate function mapping table in the STM32F40x and STM32F41x datasheets.

01: the LSI internal clock is connected to the TIM5_CH4 input for calibration purposes

10: the LSE internal clock is connected to the TIM5_CH4 input for calibration purposes

11: the RTC output event is connected to the TIM5_CH4 input for calibration purposes

Bits 5:0 Reserved, must be kept at reset value.

14.4.21 TIMx register map

TIMx registers are mapped as described in the table below:

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
0x00	TIMx_CR1	Reserved												CKD [1:0]		ARPE		CMS [1:0]		DIR		OPM		URS		UDIS		CEN															
		0 0 0 0 0 0 0 0 0 0 0 0												0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		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	TIMx_CR2	Reserved												TI1S		MMS[2:0]		CCDS		Reserved		Reserved		Reserved		Reserved		Reserved															
		0 0 0 0 0 0 0 0 0 0 0 0												0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		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	TIMx_SMCR	Reserved						ETP		ECE		ETPS [1:0]		ETF[3:0]				MSM		TS[2:0]		SMS[2:0]		Reserved		Reserved		SMS[2:0]															
		0 0 0 0 0 0 0 0 0 0 0 0						0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0				0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		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	TIMx_DIER	Reserved												TDE		COMDE		CC4DE		CC3DE		CC2DE		CC1DE		UDE		Reserved		TIE		Reserved											
		0 0 0 0 0 0 0 0 0 0 0 0												0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		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	TIMx_SR	Reserved												CC4OF		CC3OF		CC2OF		CC1OF		Reserved		TIF		Reserved		CC4IF		CC3IF		CC2IF		CC1IF		UIF							
		0 0 0 0 0 0 0 0 0 0 0 0												0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		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	TIMx_EGR	Reserved												TG		CC4G		CC3G		CC2G		CC1G		UG		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0									
		0 0 0 0 0 0 0 0 0 0 0 0												0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		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	TIMx_CCMR1 Output Compare mode	Reserved						OC2CE		OC2M [2:0]		OC2PE		OC2FE		CC2S [1:0]		OC1CE		OC1M [2:0]		OC1PE		CC1S [1:0]		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0									
		0 0 0 0 0 0 0 0 0 0 0 0						0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0													
		0 0 0 0 0 0 0 0 0 0 0 0						IC2F[3:0]		IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0											

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	TIMx_CCMR2 Output Compare mode	Reserved												OC4CE	OC4M [2:0]		OC4PE	OC4FE		CC4S [1:0]		OC3M [2:0]		OC3PE		CC3S [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		
	TIMx_CCMR2 Input Capture mode	Reserved												IC4F[3:0]	IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]		IC3PSC [1:0]		CC3S [1:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x20	TIMx_CCER	Reserved												CC4NP	CC4P		CC4E		CC3NP		CC3P		CC2NP		CC2P		CC1NP		CC1P		CC1E			
	Reset value	0	0	0	0	0	0	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	TIMx_CNT	CNT[31:16] (TIM2 and TIM5 only, reserved on the other timers)												CNT[15:0]																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x28	TIMx_PSC	Reserved												PSC[15:0]																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x2C	TIMx_ARR	ARR[31:16] (TIM2 and TIM5 only, reserved on the other timers)												ARR[15:0]																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x30	Reserved																																	
0x34	TIMx_CCR1	CCR1[31:16] (TIM2 and TIM5 only, reserved on the other timers)												CCR1[15:0]																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x38	TIMx_CCR2	CCR2[31:16] (TIM2 and TIM5 only, reserved on the other timers)												CCR2[15:0]																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x3C	TIMx_CCR3	CCR3[31:16] (TIM2 and TIM5 only, reserved on the other timers)												CCR3[15:0]																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x40	TIMx_CCR4	CCR4[31:16] (TIM2 and TIM5 only, reserved on the other timers)												CCR4[15:0]																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x44	Reserved																																	
0x48	TIMx_DCR	Reserved												DBL[4:0]				Reserved				DBA[4:0]				0 0 0 0 0 0								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x4C	TIMx_DMAR	Reserved												DMAB[15:0]																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x50	TIM2_OR	Not available												ITR1_RMP	Reserved				Reserved				IT4_RMP				0 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		
0x50	TIM5_OR	Not available												Reserved				IT4_RMP				Reserved				0 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			

Refer to [Table 1 on page 50](#) for the register boundary addresses.

15 General-purpose timers (TIM9 to TIM14)

This section applies to the whole STM32F40x and STM32F41x family, unless otherwise specified.

15.1 TIM9 to TIM14 introduction

The TIM9 to TIM14 general-purpose timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM9 to TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 15.4.12](#).

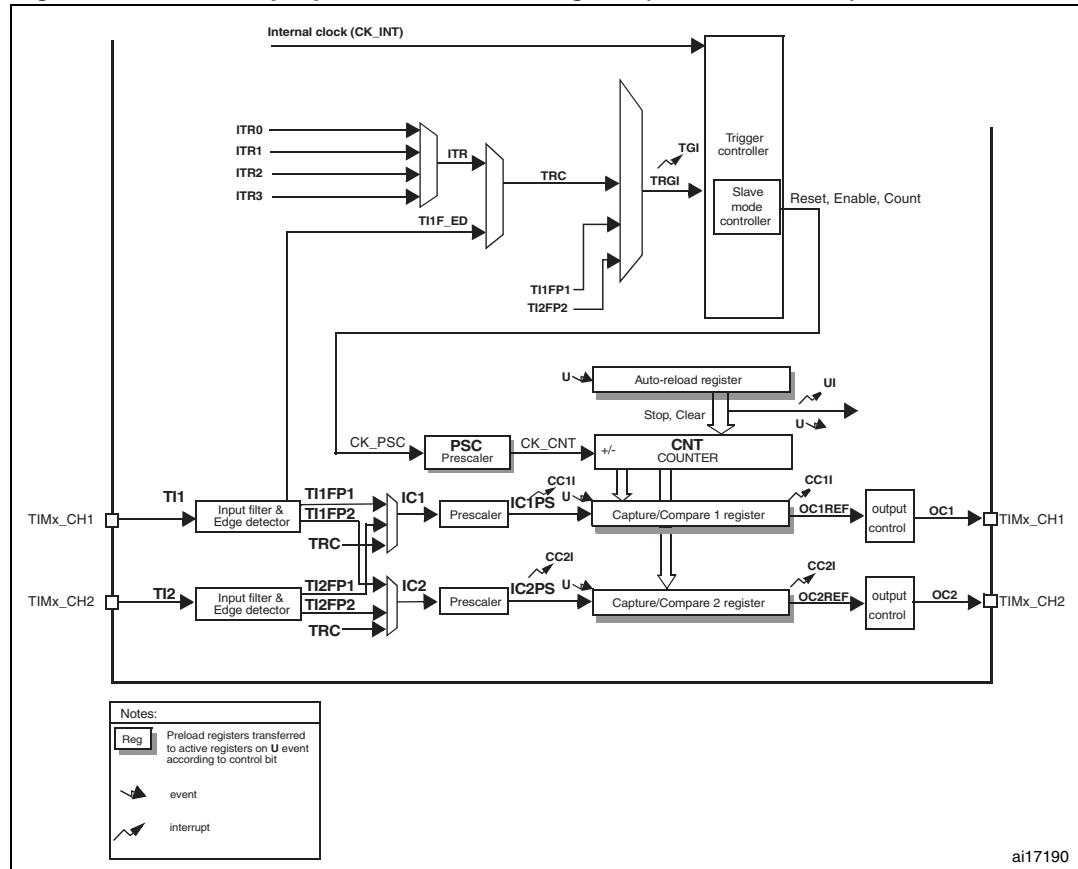
15.2 TIM9 to TIM14 main features

15.2.1 TIM9/TIM12 main features

The features of the TIM9/TIM12 general-purpose timers include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65535 (can be changed “on the fly”)
- Up to 2 independent channels for:
 - Input capture
 - Output compare
 - PWM generation (edge-aligned mode)
 - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
 - Update: counter overflow, counter initialization (by software or internal trigger)
 - Trigger event (counter start, stop, initialization or count by internal trigger)
 - Input capture
 - Output compare

Figure 153. General-purpose timer block diagram (TIM9 and TIM12)

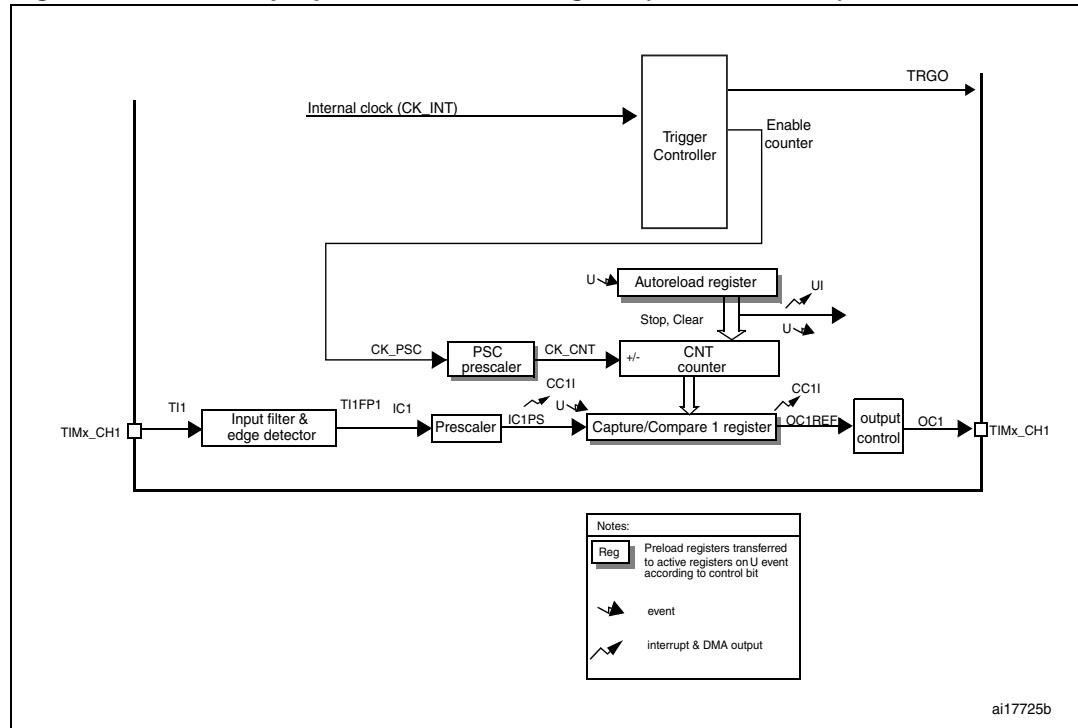


15.3 TIM10/TIM11 and TIM13/TIM14 main features

The features of general-purpose timers TIM10/TIM11 and TIM13/TIM14 include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65535 (can be changed “on the fly”)
- independent channel for:
 - Input capture
 - Output compare
 - PWM generation (edge-aligned mode)
- Interrupt generation on the following events:
 - Update: counter overflow, counter initialization (by software)
 - Input capture
 - Output compare

Figure 154. General-purpose timer block diagram (TIM10/11/13/14)



15.4 TIM9 to TIM14 functional description

15.4.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

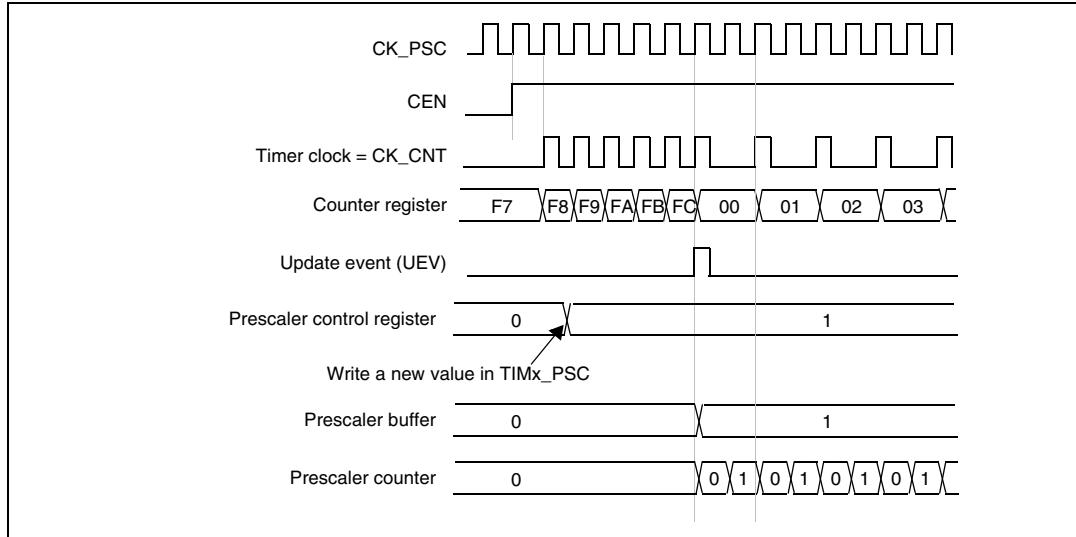
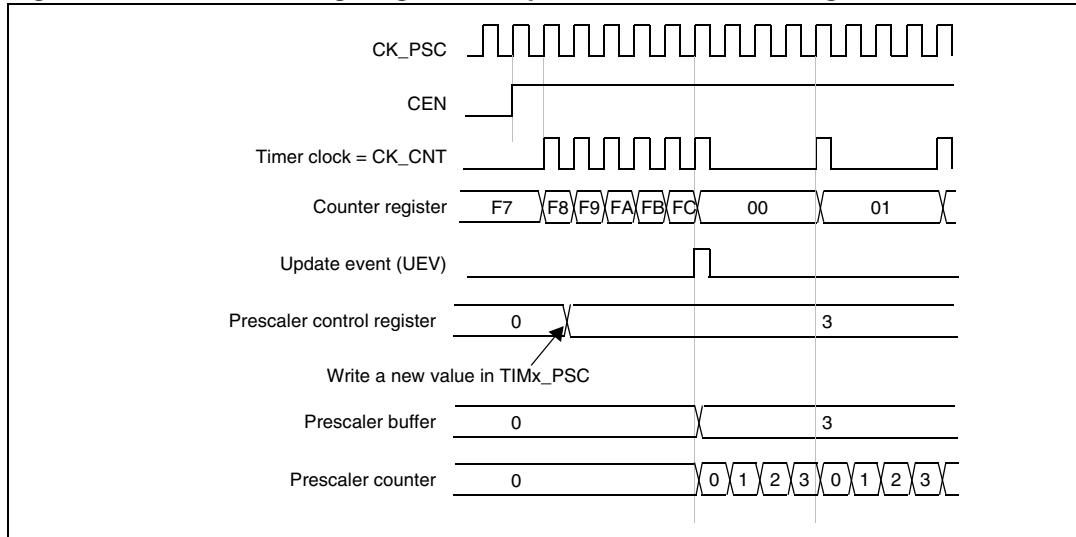
The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 156 and *Figure 157* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 155. Counter timing diagram with prescaler division change from 1 to 2**Figure 156. Counter timing diagram with prescaler division change from 1 to 4**

15.4.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM9 and TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without

setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 157. Counter timing diagram, internal clock divided by 1

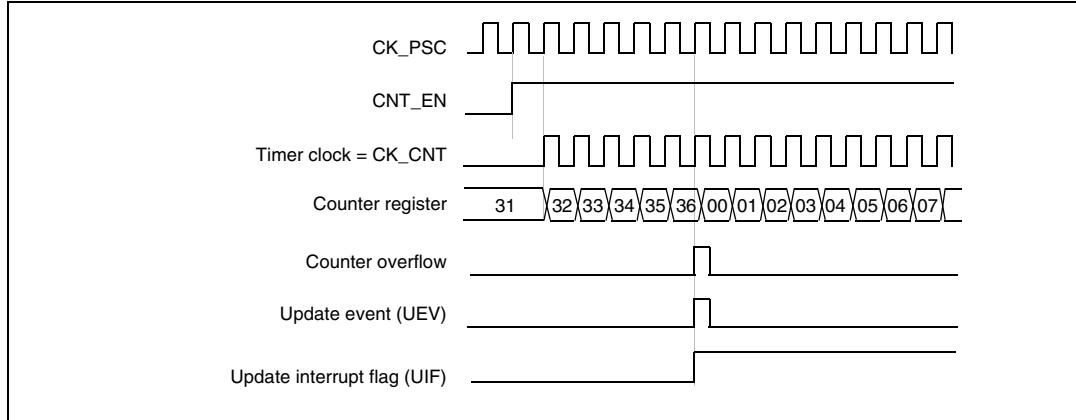


Figure 158. Counter timing diagram, internal clock divided by 2

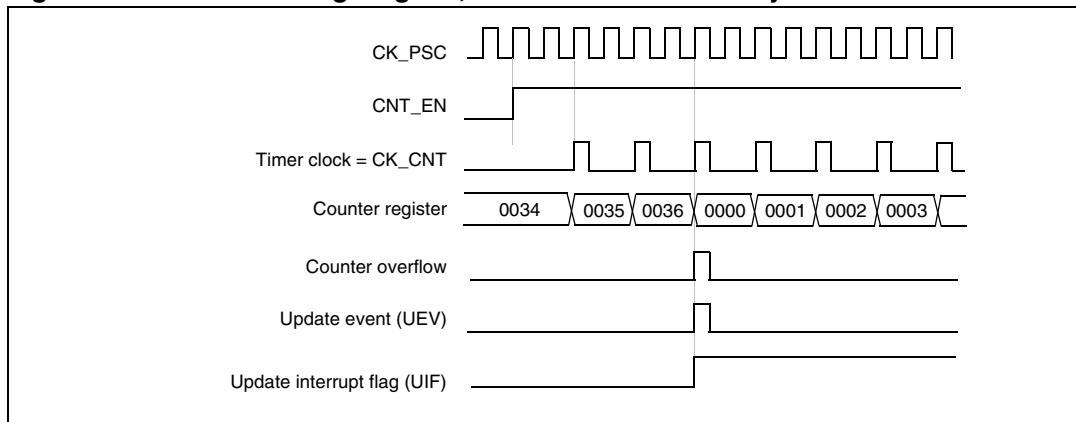


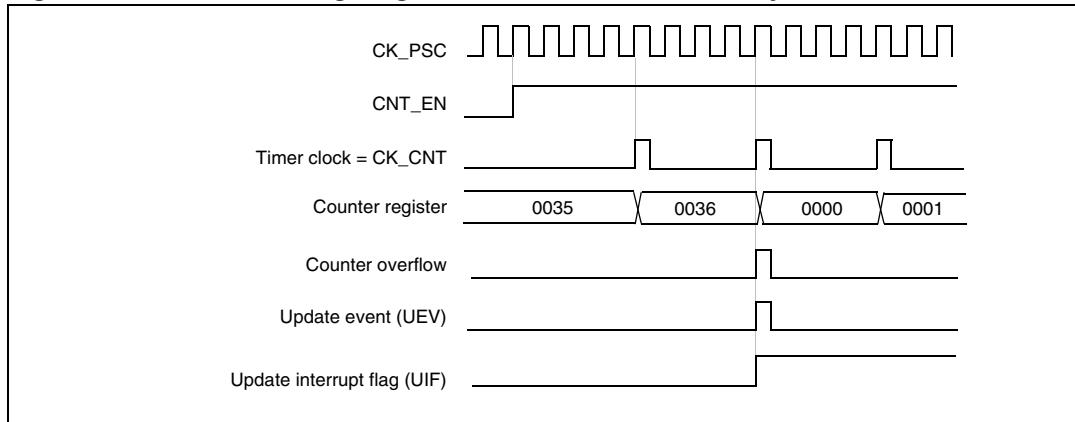
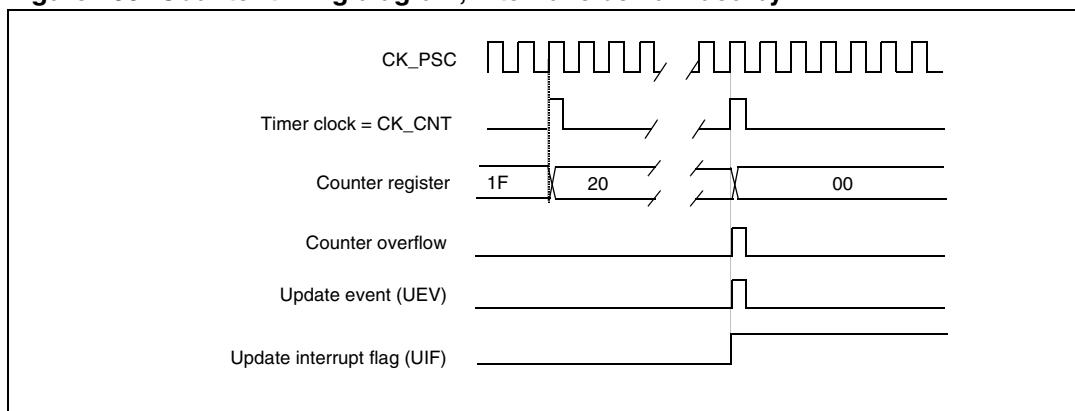
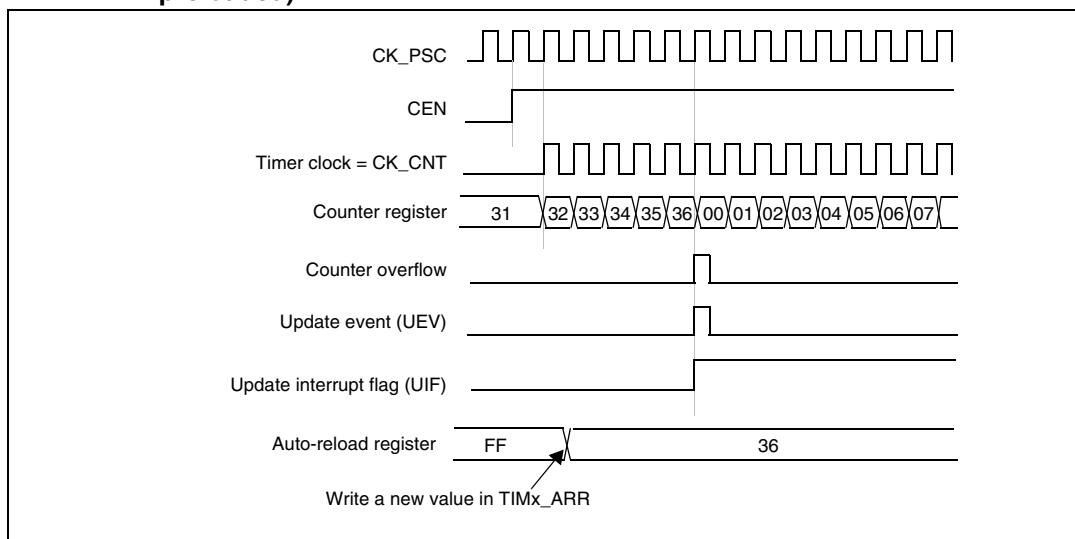
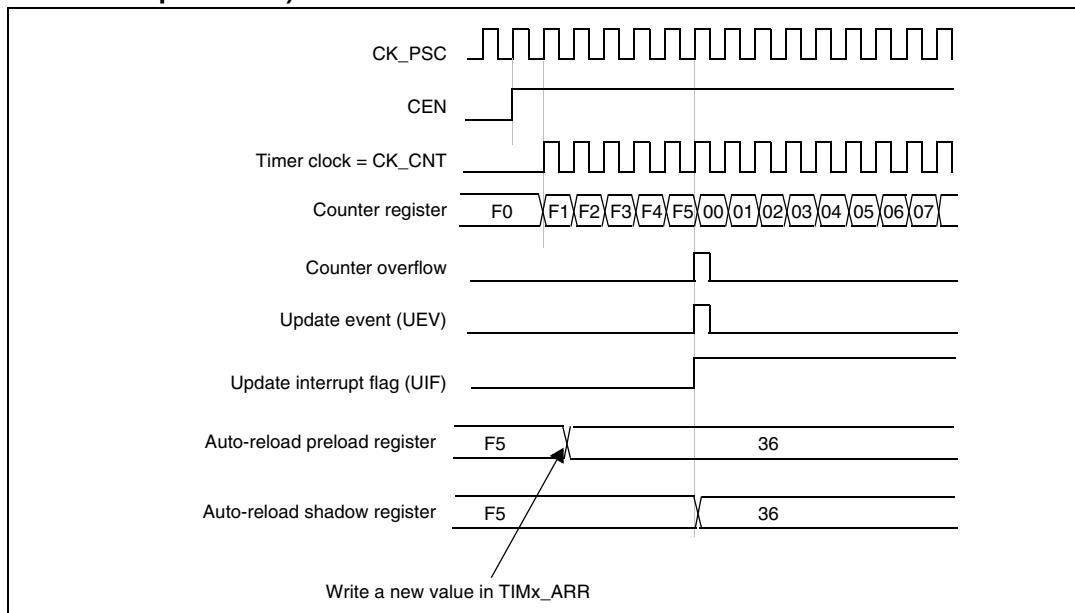
Figure 159. Counter timing diagram, internal clock divided by 4**Figure 160. Counter timing diagram, internal clock divided by N****Figure 161. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)**

Figure 162. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



15.4.3 Clock selection

The counter clock can be provided by the following clock sources:

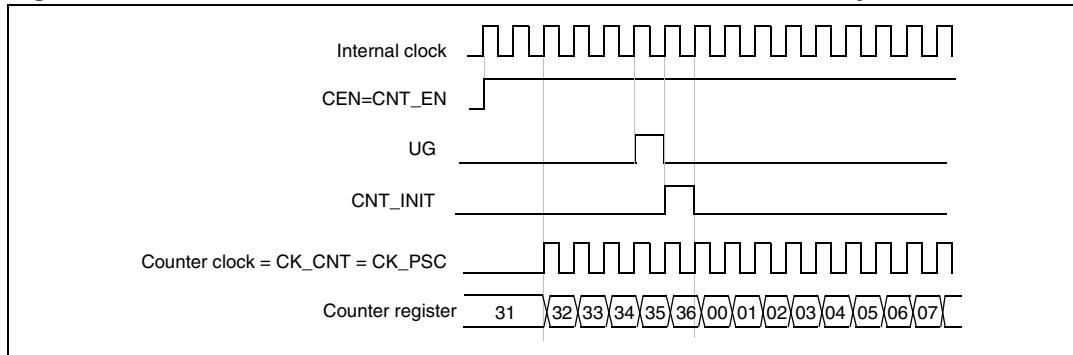
- Internal clock (CK_INT)
- External clock mode1 (for **TIM9 and TIM12**): external input pin (TIx)
- Internal trigger inputs (ITRx) (for **TIM9 and TIM12**): connecting the trigger output from another timer. Refer to [Section : Using one timer as prescaler for another](#) for more details.

Internal clock source (CK_INT)

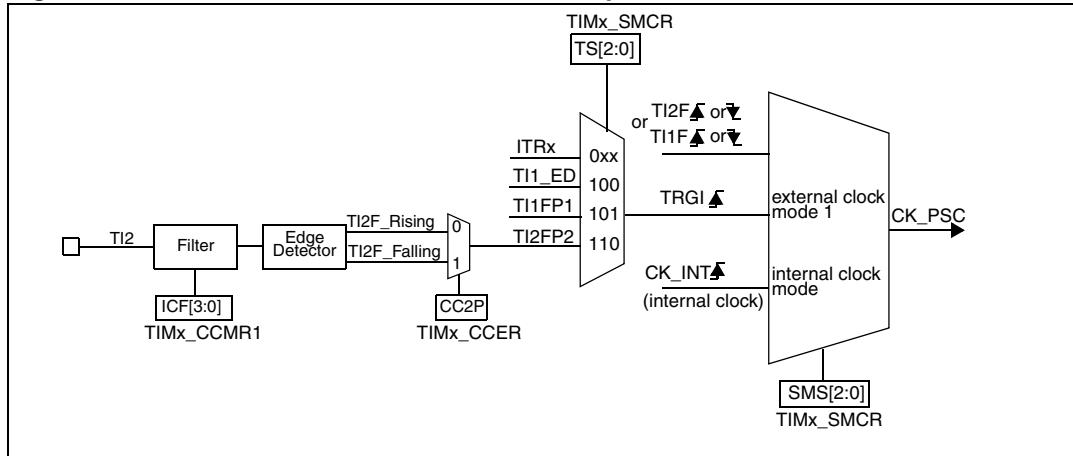
The internal clock source is the default clock source for TIM10/TIM11 and TIM13/TIM14.

For TIM9 and TIM12, the internal clock source is selected when the slave mode controller is disabled ('SMS='000'). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock CK_INT.

[Figure 163](#) shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 163. Control circuit in normal mode, internal clock divided by 1**External clock source mode 1(TIM9 and TIM12)**

This mode is selected when SMS='111' in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 164. TI2 external clock connection example

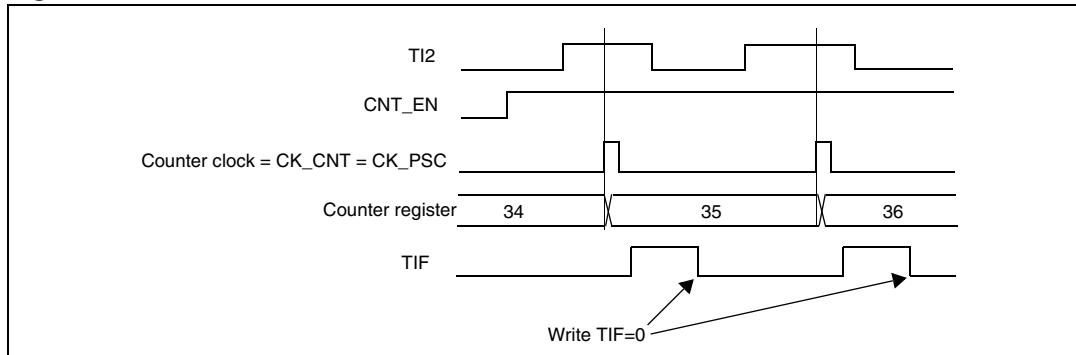
For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the **TIMx_CCMR1** register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the **TIMx_CCMR1** register (if no filter is needed, keep IC2F='0000').
3. Select the rising edge polarity by writing CC2P='0' and CC2NP='0' in the **TIMx_CCER** register.
4. Configure the timer in external clock mode 1 by writing SMS='111' in the **TIMx_SMCR** register.
5. Select TI2 as the trigger input source by writing TS='110' in the **TIMx_SMCR** register.
6. Enable the counter by writing CEN='1' in the **TIMx_CR1** register.

Note: *The capture prescaler is not used for triggering, so you don't need to configure it.*

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

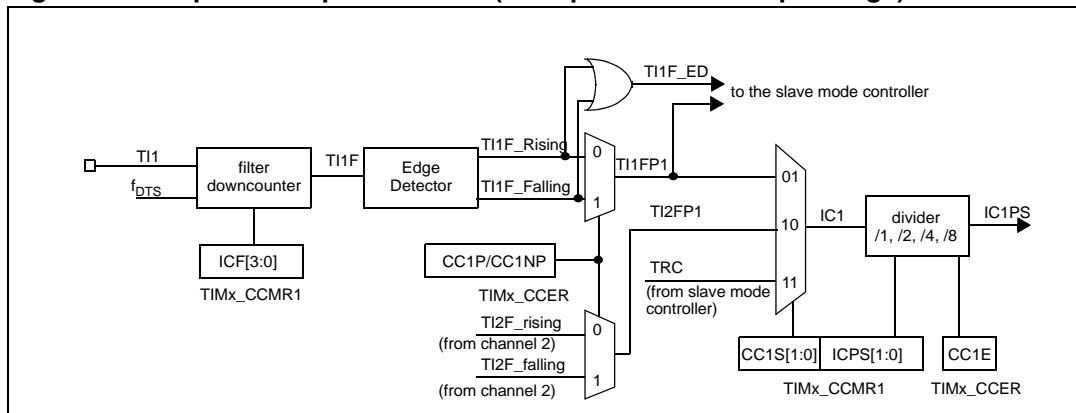
Figure 165. Control circuit in external clock mode 1

15.4.4 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 166 to Figure 168 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 166. 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 167. Capture/compare channel 1 main circuit

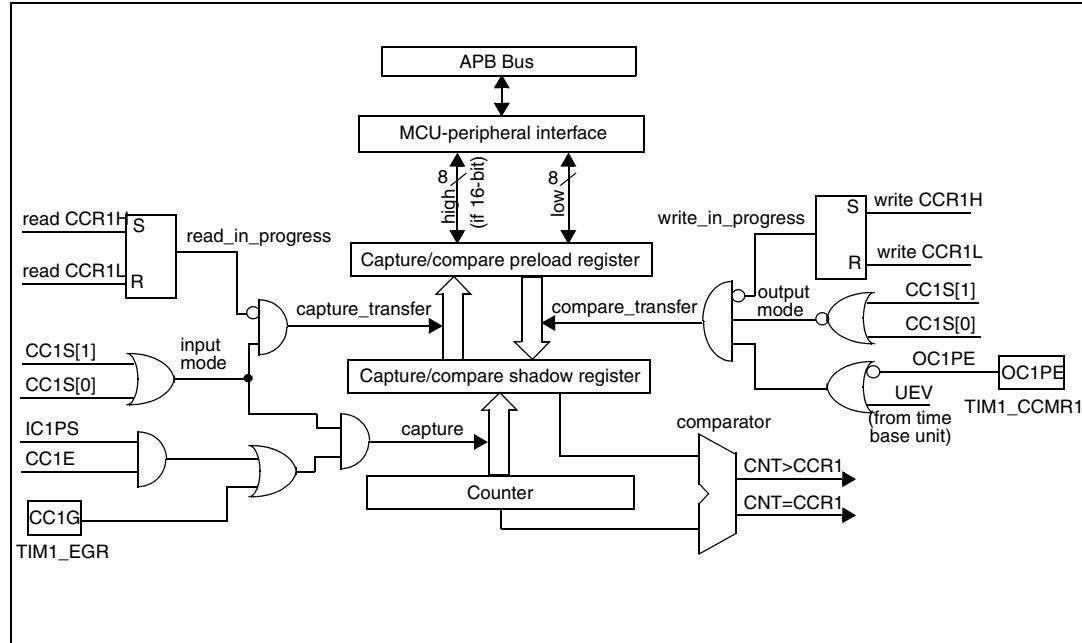
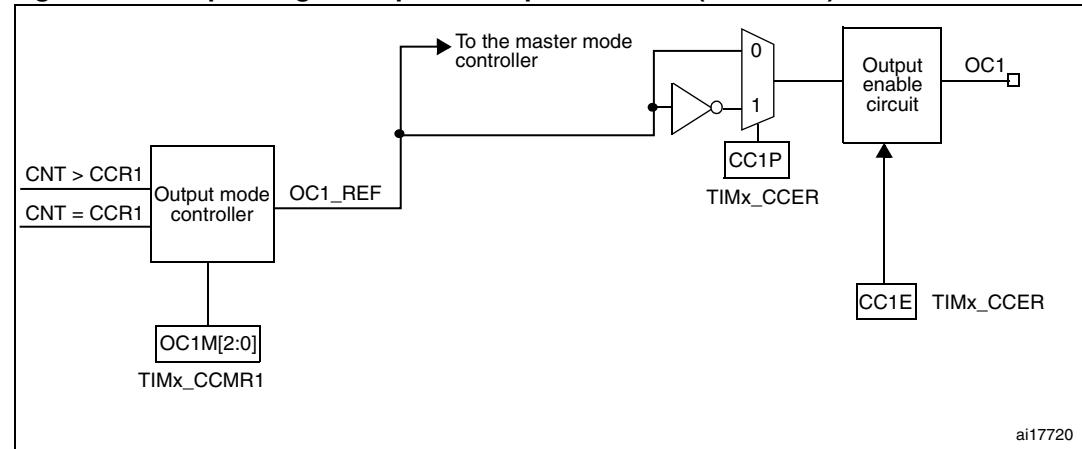


Figure 168. Output stage of capture/compare channel (channel 1)



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.

15.4.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be

cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to '01' in the TIMx_CCMR1 register. As soon as CC1S becomes different from '00', the channel is configured in input mode and the TIMx_CCR1 register becomes read-only.
2. Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to '0011' in the TIMx_CCMR1 register.
3. Select the edge of the active transition on the TI1 channel by programming CC1P and CC1NP bits to '00' in the TIMx_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: *IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.*

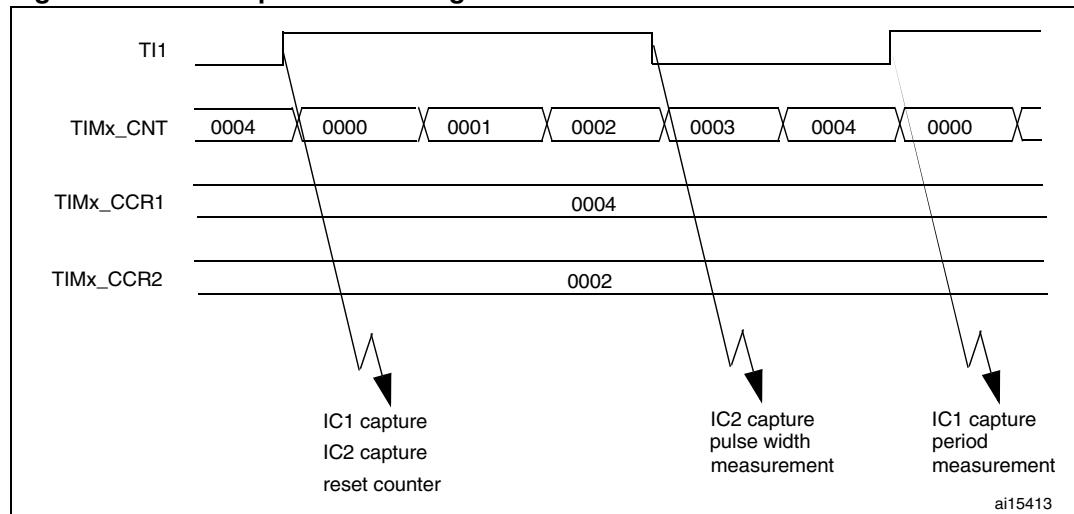
15.4.6 PWM input mode (only for TIM9/12)

This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the active input for TIMx_CCR1: write the CC1S bits to '01' in the TIMx_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): program the CC1P and CC1NP bits to '00' (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to '10' in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI2FP2 (used for capture in TIMx_CCR2): program the CC2P and CC2NP bits to '11' (active on falling edge).
5. Select the valid trigger input: write the TS bits to '101' in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to '100' in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 169. PWM input mode timing

1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

15.4.7 Forced output mode

In output mode (CCxS bits = '00' in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write '101' in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP='0' (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to '100' in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

15.4.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM='000'), be set active (OCXM='001'), be set inactive (OCXM='010') or can toggle (OCXM='011') on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

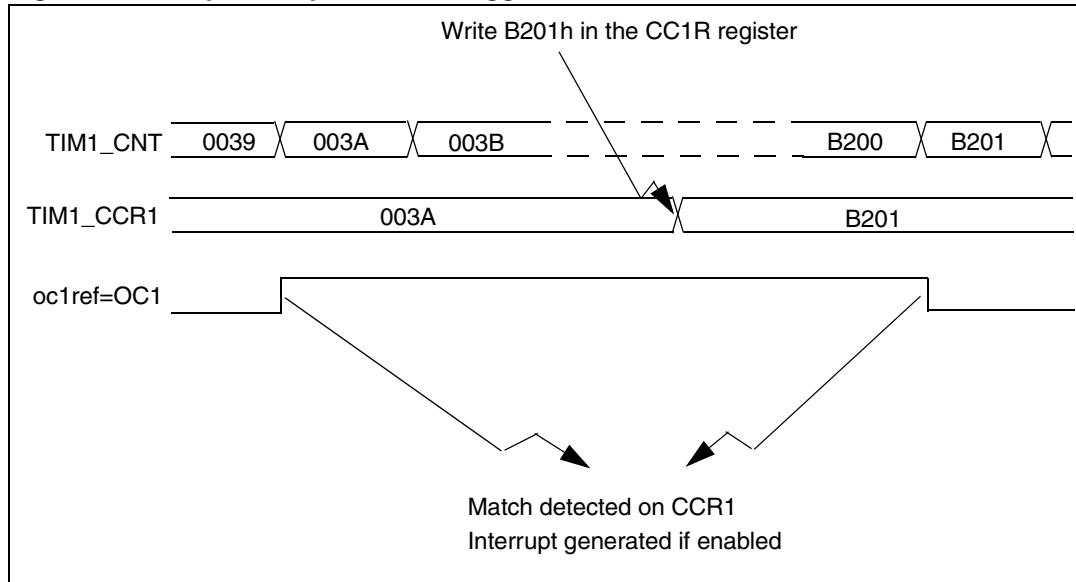
The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
 - Write OCxM = '011' to toggle OCx output pin when CNT matches CCRx
 - Write OCxPE = '0' to disable preload register
 - Write CCxP = '0' to select active high polarity
 - Write CCxE = '1' to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 170](#).

Figure 170. Output compare mode, toggle on OC1.

15.4.9 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

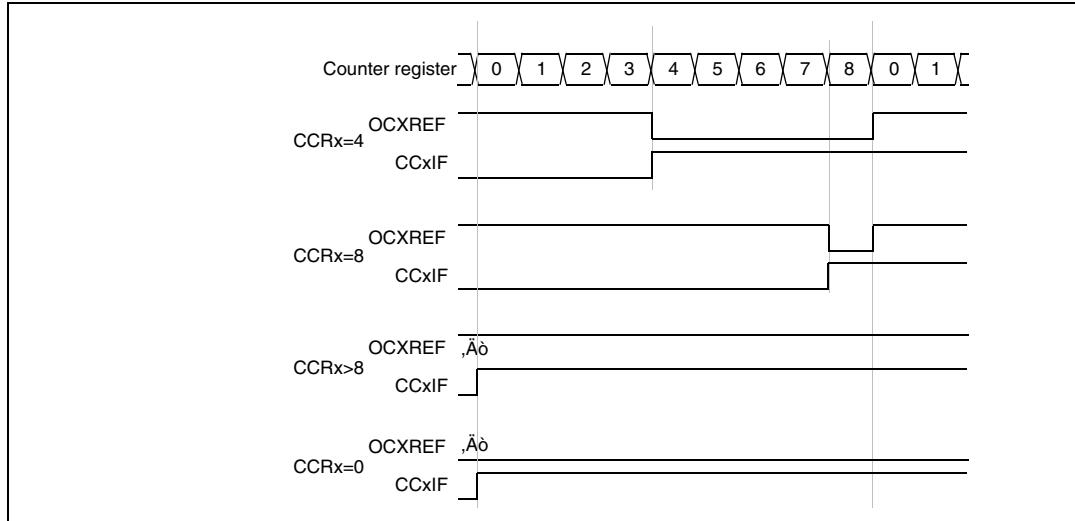
The OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. The OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether $\text{TIMx_CNT} \leq \text{TIMx_CCRx}$.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

PWM edge-aligned mode

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as $\text{TIMx_CNT} < \text{TIMx_CCRx}$ else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at '1'. If the compare value is 0 then OCxRef is held at '0'. [Figure 171](#) shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 171. Edge-aligned PWM waveforms (ARR=8)

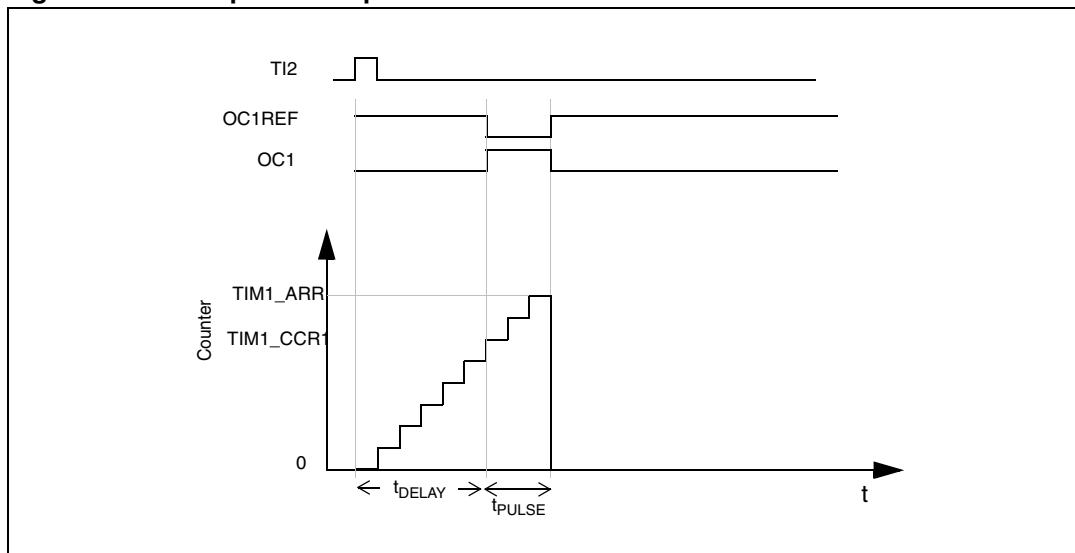
15.4.10 One-pulse mode (only for TIM9/12)

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

$$\text{CNT} < \text{CCRx} \leq \text{ARR} \text{ (in particular, } 0 < \text{CCRx})$$

Figure 172. Example of one pulse mode.

For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Use TI2FP2 as trigger 1:

1. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
2. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP = '0' in the TIMx_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
4. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M='111' in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

You only want 1 pulse (Single mode), so you write '1' in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

15.4.11 TIM9/12 external trigger synchronization

The TIM9/12 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.

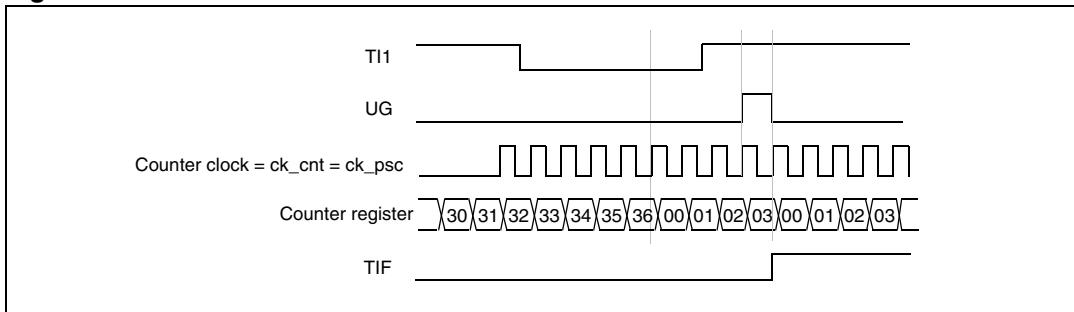
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

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. Program CC1P and CC1NP to '00' in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS='100' in TIMx_SMCR register. Select TI1 as the input source by writing TS='101' in TIMx_SMCR register.
3. Start the counter by writing CEN='1' in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 173. Control circuit in reset mode



Slave mode: Gated mode

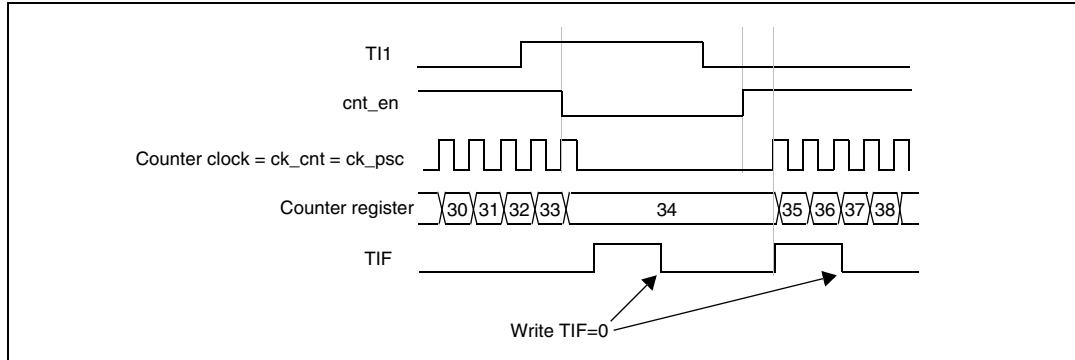
The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S='01' in TIMx_CCMR1 register. Program CC1P='1' and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS='101' in TIMx_SMCR register. Select TI1 as the input source by writing TS='101' in TIMx_SMCR register.
3. Enable the counter by writing CEN='1' in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN='0', whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 174. Control circuit in gated mode**Slave mode: Trigger mode**

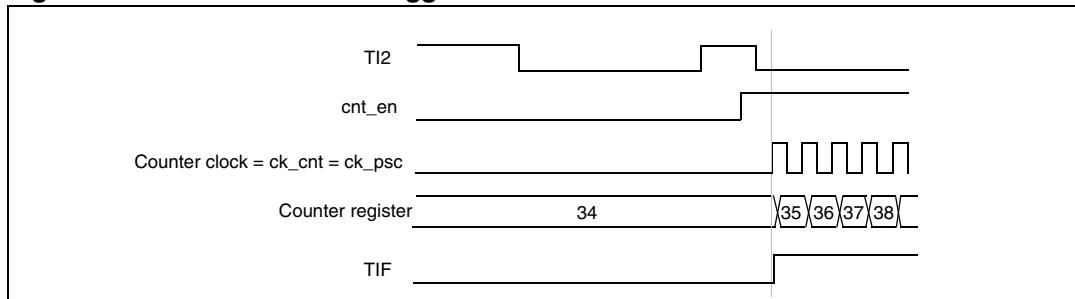
The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on **TI2** input:

1. Configure the channel 2 to detect rising edges on **TI2**. Configure the input filter duration (in this example, we don't need any filter, so we keep **IC2F='0000'**). The capture prescaler is not used for triggering, so you don't need to configure it. The **CC2S** bits are configured to select the input capture source only, **CC2S='01'** in **TIMx_CCMR1** register. Program **CC2P='1'** and **CC2NP='0'** in **TIMx_CCER** register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing **SMS='110'** in **TIMx_SMCR** register. Select **TI2** as the input source by writing **TS='110'** in **TIMx_SMCR** register.

When a rising edge occurs on **TI2**, the counter starts counting on the internal clock and the **TIF** flag is set.

The delay between the rising edge on **TI2** and the actual start of the counter is due to the resynchronization circuit on **TI2** input.

Figure 175. Control circuit in trigger mode

15.4.12 Timer synchronization (TIM9/12)

The TIM timers are linked together internally for timer synchronization or chaining. Refer to [Section 14.3.15: Timer synchronization on page 391](#) for details.

15.4.13 Debug mode

When the microcontroller enters debug mode (Cortex™-M4F 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 32.16.2: Debug support for timers, watchdog, bxCAN and I²C](#).

15.5 TIM9 and TIM12 registers

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

15.5.1 TIM9/12 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved				CKD[1:0]		ARPE		reserved		OPM		URS		UDIS		CEN	
				rw	rw	rw					rw	rw	rw	rw		rw	

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

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (TIx),

- 00: $t_{DTS} = t_{CK_INT}$
- 01: $t_{DTS} = 2 \times t_{CK_INT}$
- 10: $t_{DTS} = 4 \times t_{CK_INT}$
- 11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

- 0: TIMx_ARR register is not buffered.
- 1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

- 0: Counter is not stopped on the update event
- 1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

- 0: Any of the following events generates an update interrupt if enabled:

- Counter overflow
- Setting the UG bit

1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update event (UEV) generation.

- 0: UEV enabled. An UEV is generated by one of the following events:

- Counter overflow
- Setting the UG bit

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

- 0: Counter disabled
- 1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

15.5.2 TIM9/12 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

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

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

Bits 6:4 **MMS**: Master mode selection

These bits are used to select the information to be sent in Master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit in the TIMx_EGR register is used as the trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as the trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between the CEN control bit and the trigger input when configured in Gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).

010: **Update** - The update event is selected as the trigger output (TRGO). For instance a master timer can be used as a prescaler for a slave timer.

011: **Compare pulse** - The trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurs. (TRGO).

100: **Compare** - OC1REF signal is used as the trigger output (TRGO).

101: **Compare** - OC2REF signal is used as the trigger output (TRGO).

110: Reserved

111: Reserved

Bits 3:0 Reserved, must be kept at reset value.

15.5.3 TIM9/12 slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								MSM	TS[2:0]		Res.	SMS[2:0]		rw	rw	rw
rw		rw		rw		rw										

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

Bit 7 **MSM:** Master/Slave mode

- 0: No action
- 1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful in order to synchronize several timers on a single external event.

Bits 6:4 **TS:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

- 000: Internal Trigger 0 (ITR0)
- 001: Internal Trigger 1 (ITR1)
- 010: Internal Trigger 2 (ITR2)
- 011: Internal Trigger 3 (ITR3)
- 100: TI1 Edge Detector (TI1F_ED)
- 101: Filtered Timer Input 1 (TI1FP1)
- 110: Filtered Timer Input 2 (TI2FP2)
- 111: Reserved.

See [Table 62: TIMx internal trigger connection on page 437](#) for more details on the meaning of ITRx for each timer.

Note: These bits must be changed only when they are not used (e.g. when SMS='000') to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMS:** Slave mode selection

When external signals are selected, the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input control register and Control register descriptions).

- 000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock
- 001: Reserved
- 010: Reserved
- 011: Reserved
- 100: Reset mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers
- 101: Gated mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Counter starts and stops are both controlled
- 110: Trigger mode - The counter starts on a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled
- 111: External clock mode 1 - Rising edges of the selected trigger (TRGI) clock the counter

Note: The Gated mode must not be used if TI1F_ED is selected as the trigger input (TS='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the Gated mode checks the level of the trigger signal.

Table 62. TIMx internal trigger connection

Slave TIM	ITR0 (TS = '000')	ITR1 (TS = '001')	ITR2 (TS = '010')	ITR3 (TS = '011')
TIM2	TIM1	TIM8	TIM3	TIM4
TIM3	TIM1	TIM2	TIM5	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8
TIM9	TIM2	TIM3	TIM10	TIM11
TIM12	TIM4	TIM5	TIM13	TIM14

15.5.4 TIM9/12 Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TIE	Res		CC2IE	CC1IE	UIE		
								rw			rw	rw	rw		

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

Bit 6 **TIE**: Trigger interrupt enable

- 0: Trigger interrupt disabled.
- 1: Trigger interrupt enabled.

Bit 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable

- 0: CC2 interrupt disabled.
- 1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

- 0: CC1 interrupt disabled.
- 1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable

- 0: Update interrupt disabled.
- 1: Update interrupt enabled.

15.5.5 TIM9/12 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved				CC2OF	CC1OF	Reserved	TIF	Reserved			CC2IF	CC1IF	UIF	rc_w0	rc_w0	rc_w0

Bit 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

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

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output:

This flag is set by hardware when the counter matches the compare value. It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.

If channel CC1 is configured as input:

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred.

1: The counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

Bit 0 UIF: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow and if UDIS='0' in the TIMx_CR1 register.

- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS='0' and UDIS='0' in the TIMx_CR1 register.

- When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS='0' and UDIS='0' in the TIMx_CR1 register.

15.5.6 TIM9/12 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
									TG				CC2G	CC1G	UG
									w				w	w	w

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

Bit 6 TG: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 CC2G: Capture/compare 2 generation

refer to CC1G description

Bit 1 CC1G: Capture/compare 1 generation

This bit is set by software to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

the CC1IF flag is set, the corresponding interrupt is sent if enabled.

If channel CC1 is configured as input:

The current counter value is captured in the TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 UG: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

15.5.7 TIM9/12 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 CC_xS bits. All the other bits in this register have different functions in input and output modes. For a given bit, OC_{xx} describes its function when the channel is configured in output mode, IC_{xx} describes its function when the channel is configured in input mode. So you must take care that the same bit can have different meanings for the input stage and the output stage.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]			
	IC2F[3:0]			IC2PSC[1:0]				IC1F[3:0]			IC1PSC[1:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **OC2M[2:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **OC1M:** Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas the active levels of OC1 and OC1N depend on the CC1P and CC1NP bits, respectively.

000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

001: Set channel 1 to active level on match. The OC1REF signal is forced high when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

010: Set channel 1 to inactive level on match. The OC1REF signal is forced low when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1

100: Force inactive level - OC1REF is forced low

101: Force active level - OC1REF is forced high

110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else it is inactive. In downcounting, channel 1 is inactive (OC1REF='0) as long as TIMx_CNT>TIMx_CCR1, else it is active (OC1REF='1')

111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else it is active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else it is inactive.

Note: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC1PE:** Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken into account immediately

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

Note: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in the TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE:** Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger input on the CC output.
0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles

1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S:** Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F**: Input capture 1 filter

This bitfield defines the frequency used to sample the TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}	1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
0001: $f_{SAMPLING}=f_{CK_INT}$, N=2	1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
0010: $f_{SAMPLING}=f_{CK_INT}$, N=4	1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
0011: $f_{SAMPLING}=f_{CK_INT}$, N=8	1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6	1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8	1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6	1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8	1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: In the current silicon revision, f_{DTS} is replaced in the formula by CK_INT when ICxF[3:0]= 1, 2 or 3.

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

15.5.8 TIM9/12 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

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

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity
refer to CC1NP description

Bits 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity
refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity
CC1 channel configured as output: CC1NP must be kept cleared
CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity (refer to CC1P description).

Bits 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

CC1 channel configured as output:

0: OC1 active high.
1: OC1 active low.

CC1 channel configured as input:

CC1NP/CC1P bits select TI1FP1 and TI2FP1 polarity for trigger or capture operations.
00: noninverted/rising edge
Circuit is sensitive to TIxFP1 rising edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode, encoder mode).
01: inverted/falling edge
Circuit is sensitive to TIxFP1 falling edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is inverted (trigger in gated mode, encoder mode).
10: reserved, do not use this configuration.

Note: 11: noninverted/both edges

Circuit is sensitive to both TIxFP1 rising and falling edges (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode). This configuration must not be used for encoder mode.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

CC1 channel configured as output:

0: Off - OC1 is not active.
1: On - OC1 signal is output on the corresponding output pin.

CC1 channel configured as input:

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled.
1: Capture enabled.

Table 63. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled ($OCx='0'$, $OCx_EN='0'$)
1	$OCx=OCxREF + \text{Polarity}$, $OCx_EN='1'$

Note: The states of the external I/O pins connected to the standard OCx channels depend on the state of the OCx channel and on the GPIO registers.

15.5.9 TIM9/12 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bits 15:0 **CNT[15:0]**: Counter value

15.5.10 TIM9/12 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.

15.5.11 TIM9/12 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to the [Section 15.4.1: Time-base unit on page 417](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

15.5.12 TIM9/12 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded into the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (OC1PE bit). Else the preload value is copied into the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signaled on the OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

15.5.13 TIM9/12 capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded into the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (OC2PE bit). Else the preload value is copied into the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signalled on the OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

15.5.14 TIM9/12 register map

TIM9/12 registers are mapped as 16-bit addressable registers as described below:

Table 64. TIM9/12 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																CKD [1:0]	ARPE	Reserved	Reserved	Reserved	OPM	URS	UDIS	CEN							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x04	TIMx_CR2	Reserved																MMS[2:0]	0	0	0	0	0	0	0	0	0	0	0				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 64. TIM9/12 register map and reset values (continued)

Refer to [Table 1 on page 50](#) for the register boundary addresses.

15.6 TIM10/11/13/14 registers

15.6.1 TIM10/11/13/14 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

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

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

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),

- 00: $t_{DTS} = t_{CK_INT}$
- 01: $t_{DTS} = 2 \times t_{CK_INT}$
- 10: $t_{DTS} = 4 \times t_{CK_INT}$
- 11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

- 0: TIMx_ARR register is not buffered
- 1: TIMx_ARR register is buffered

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the update interrupt (UEV) sources.

0: Any of the following events generate an UEV if enabled:

- Counter overflow
- Setting the UG bit

1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.

0: UEV enabled. An UEV is generated by one of the following events:

- Counter overflow
- Setting the UG bit.

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

- 0: Counter disabled
- 1: Counter enabled

15.6.2 TIM10/11/13/14 Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1IE	UIE
														rw	rw

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

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

- 0: CC1 interrupt disabled
- 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

- 0: Update interrupt disabled
- 1: Update interrupt enabled

15.6.3 TIM10/11/13/14 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved							CC1OF	Reserved							CC1IF	UIF
							rc_w0								rc_w0	rc_w0

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

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output:

This flag is set by hardware when the counter matches the compare value. It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.

If channel CC1 is configured as input:

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred.

1: The counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

Bit 0 UIF: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow and if UDIS='0' in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS='0' and UDIS='0' in the TIMx_CR1 register.

15.6.4 TIM10/11/13/14 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1G	UG
w														w	w

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

Bit 1 CC1G: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 UG: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

15.6.5 TIM10/11/13/14 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OC1M[2:0]				OC1PE	OC1FE	CC1S[1:0]	
Reserved								IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

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

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on CC1P bit.

000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

011: Toggle - OC1REF toggles when TIMx_CNT = TIMx_CCR1.

100: Force inactive level - OC1REF is forced low.

101: Force active level - OC1REF is forced high.

110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active.

Note: In PWM mode 1 or 2, the OCREF level changes when the result of the comparison changes or when the output compare mode switches from frozen to PWM mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10:

11:

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

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

Bits 7:4 **IC1F**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}	1000: $f_{SAMPLING} = f_{DTS}/8, N=6$
0001: $f_{SAMPLING} = f_{CK_INT}, N=2$	1001: $f_{SAMPLING} = f_{DTS}/8, N=8$
0010: $f_{SAMPLING} = f_{CK_INT}, N=4$	1010: $f_{SAMPLING} = f_{DTS}/16, N=5$
0011: $f_{SAMPLING} = f_{CK_INT}, N=8$	1011: $f_{SAMPLING} = f_{DTS}/16, N=6$
0100: $f_{SAMPLING} = f_{DTS}/2, N=6$	1100: $f_{SAMPLING} = f_{DTS}/16, N=8$
0101: $f_{SAMPLING} = f_{DTS}/2, N=8$	1101: $f_{SAMPLING} = f_{DTS}/32, N=5$
0110: $f_{SAMPLING} = f_{DTS}/4, N=6$	1110: $f_{SAMPLING} = f_{DTS}/32, N=6$
0111: $f_{SAMPLING} = f_{DTS}/4, N=8$	1111: $f_{SAMPLING} = f_{DTS}/32, N=8$

Note: In current silicon revision, f_{DTS} is replaced in the formula by CK_INT when ICxF[3:0]= 1, 2 or 3.

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).

The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: Reserved

11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

15.6.6 TIM10/11/13/14 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CC1NP	Res.	CC1P	CC1E		
										rw		rw	rw		

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

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared.

CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define TI1FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

CC1 channel configured as output:

0: OC1 active high

1: OC1 active low

CC1 channel configured as input:

The CC1P bit selects TI1FP1 and TI2FP1 polarity for trigger or capture operations.

00: noninverted/rising edge

Circuit is sensitive to TI1FP1 rising edge (capture mode), TI1FP1 is not inverted.

01: inverted/falling edge

Circuit is sensitive to TI1FP1 falling edge (capture mode), TI1FP1 is inverted.

10: reserved, do not use this configuration.

11: noninverted/both edges

Circuit is sensitive to both TI1FP1 rising and falling edges (capture mode), TI1FP1 is not inverted.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

CC1 channel configured as output:

0: Off - OC1 is not active

1: On - OC1 signal is output on the corresponding output pin

CC1 channel configured as input:

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

0: Capture disabled

1: Capture enabled

Table 65. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output Disabled (OCx='0', OCx_EN='0')
1	OCx=OCxREF + Polarity, OCx_EN='1'

Note:

The state of the external I/O pins connected to the standard OCx channels depends on the OCx channel state and the GPIO registers.

15.6.7 TIM10/11/13/14 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]**: Counter value

15.6.8 TIM10/11/13/14 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to $f_{CK_PSC} / (PSC[15:0] + 1)$.

PSC contains the value to be loaded in the active prescaler register at each update event.

15.6.9 TIM10/11/13/14 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to [Section 15.4.1: Time-base unit on page 417](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

15.6.10 TIM10/11/13/14 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

15.6.11 TIM11 option register 1 (TIM11_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TI1_RMP	
														rw	

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

Bits 1:0 **TI1_RMP**: TIM11 Input 1 remapping capability

Set and cleared by software.

00,01,11: TIM11 Channel1 is connected to the GPIO (refer to the Alternate function mapping table in the datasheets).

10: HSE_RTC clock (HSE divided by programmable prescaler) is connected to the TIM11_CH1 input for measurement purposes

15.6.12 TIM10/11/13/14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

Table 66. TIM10/11/13/14 register map and reset values

Refer to [Table 1 on page 50](#) for the register boundary addresses.

16 Basic timers (TIM6&TIM7)

16.1 TIM6&TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

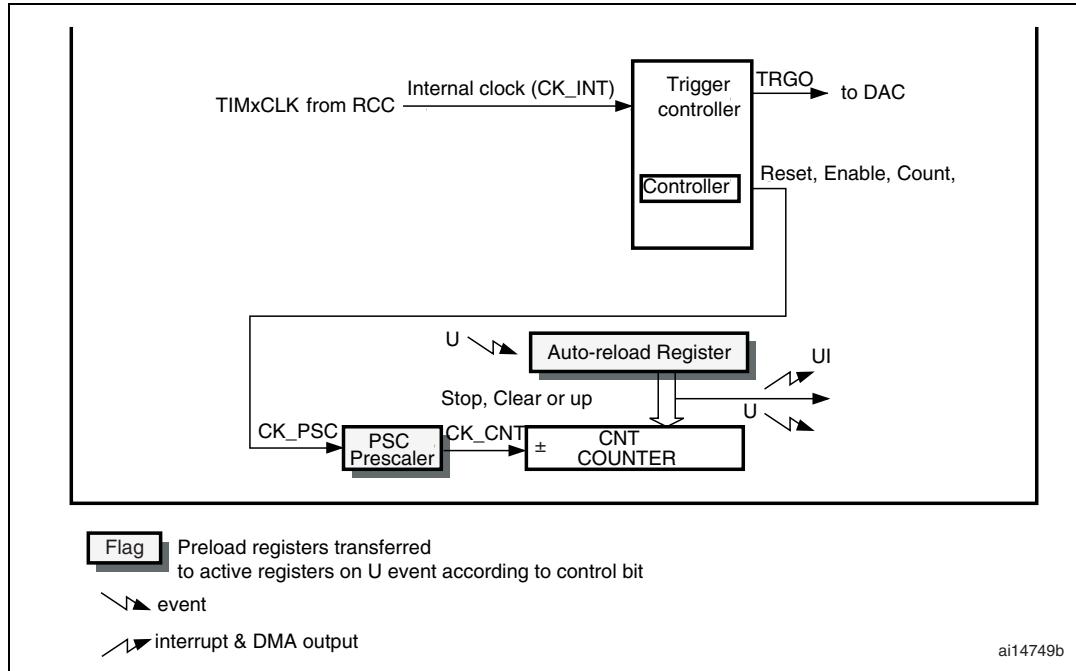
The timers are completely independent, and do not share any resources.

16.2 TIM6&TIM7 main features

Basic timer (TIM6&TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 176. Basic timer block diagram



16.3 TIM6&TIM7 functional description

16.3.1 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

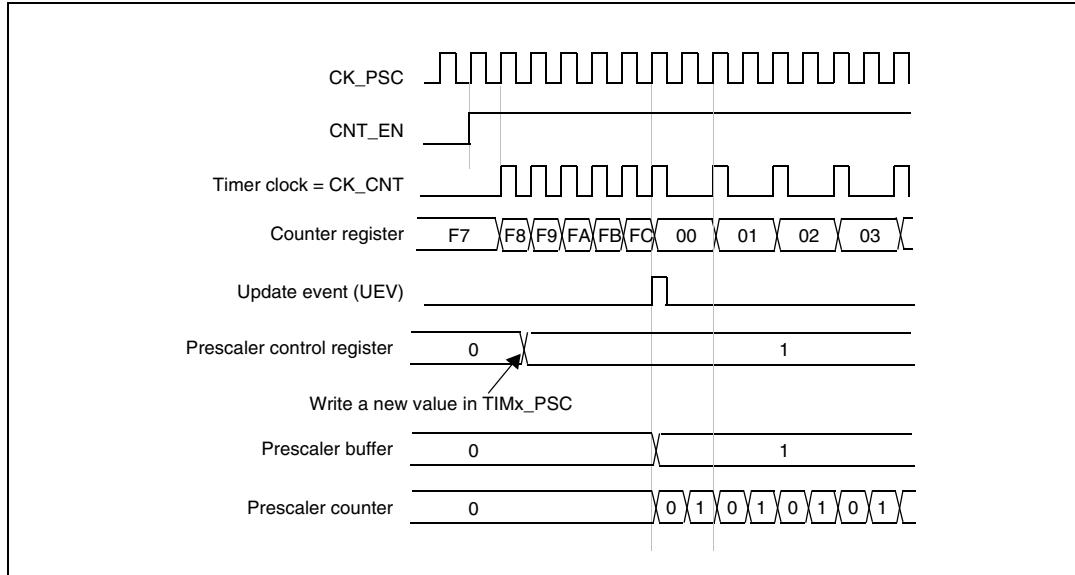
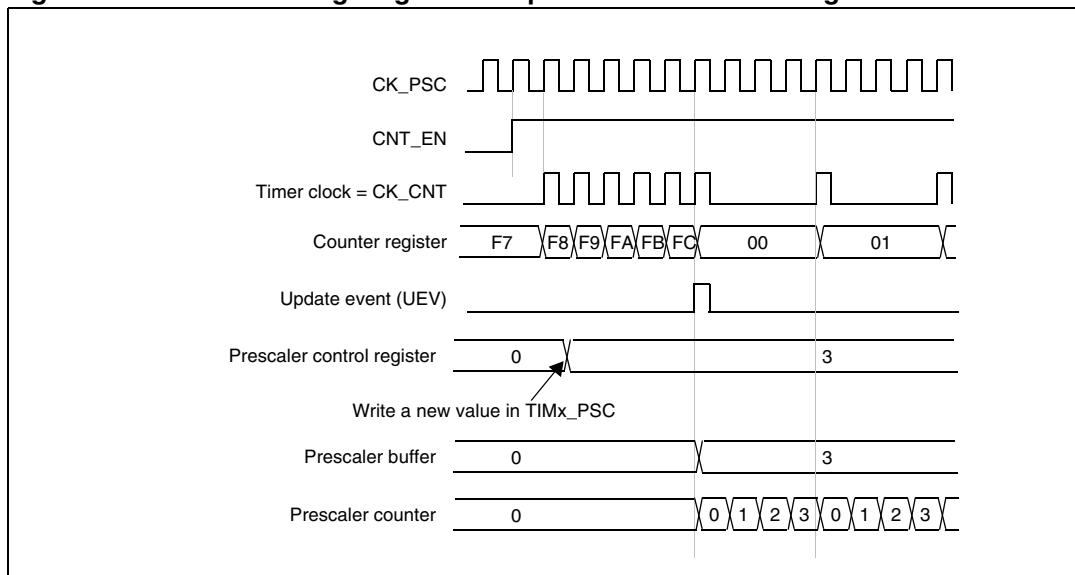
The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 177 and *Figure 178* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 177. Counter timing diagram with prescaler division change from 1 to 2**Figure 178. Counter timing diagram with prescaler division change from 1 to 4**

16.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be 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 179. Counter timing diagram, internal clock divided by 1

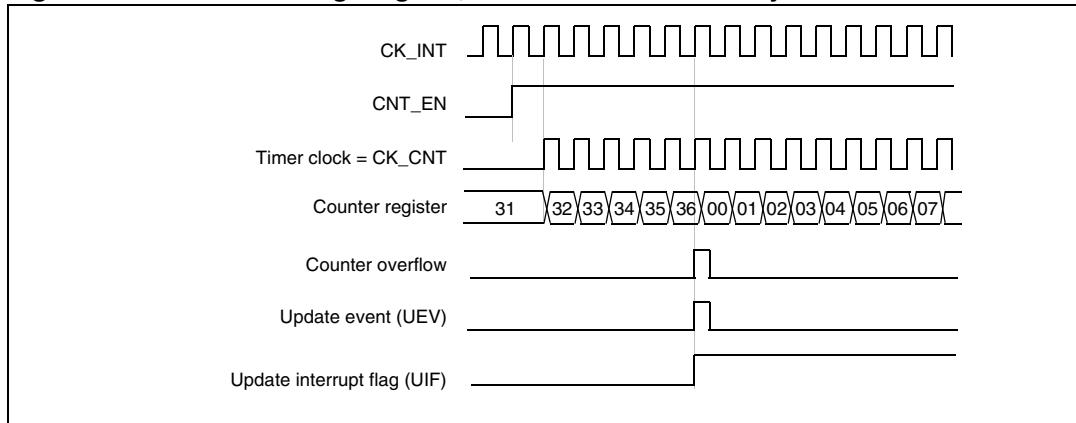


Figure 180. Counter timing diagram, internal clock divided by 2

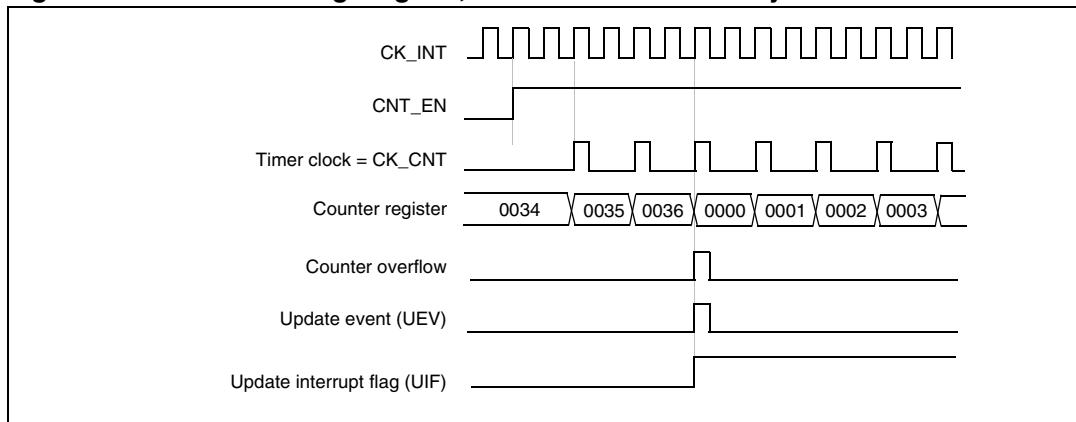


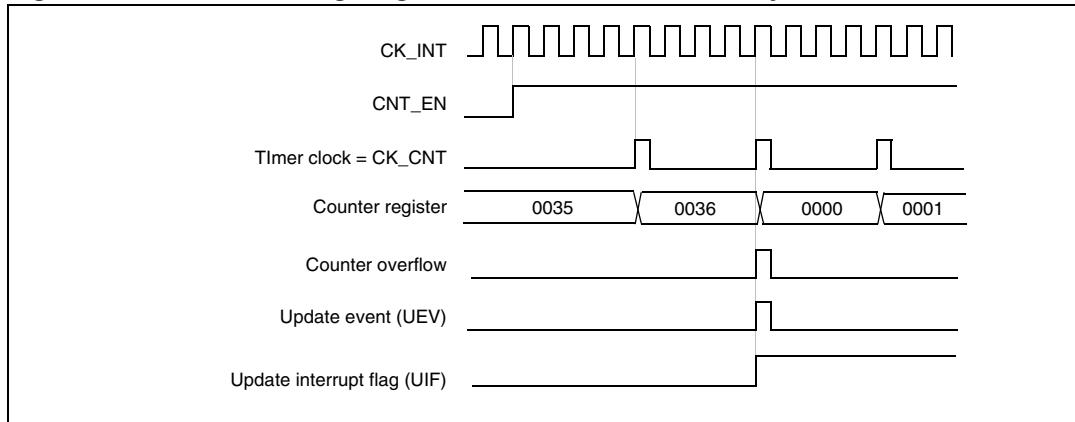
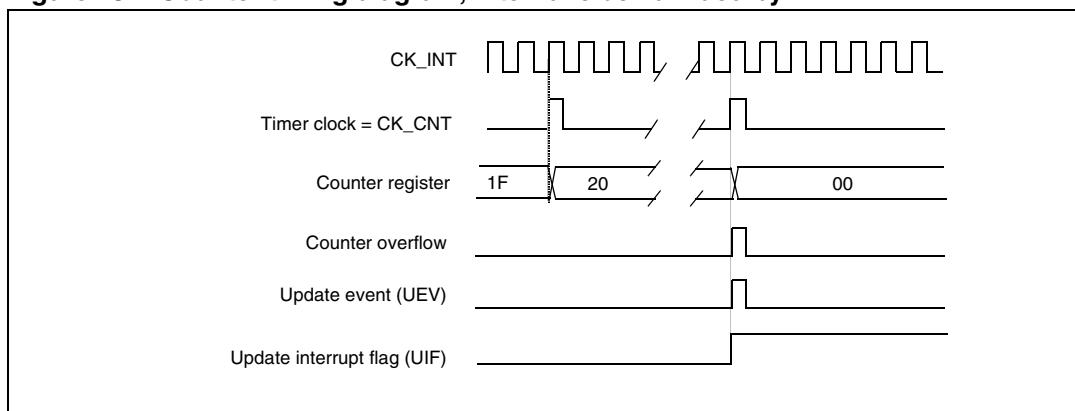
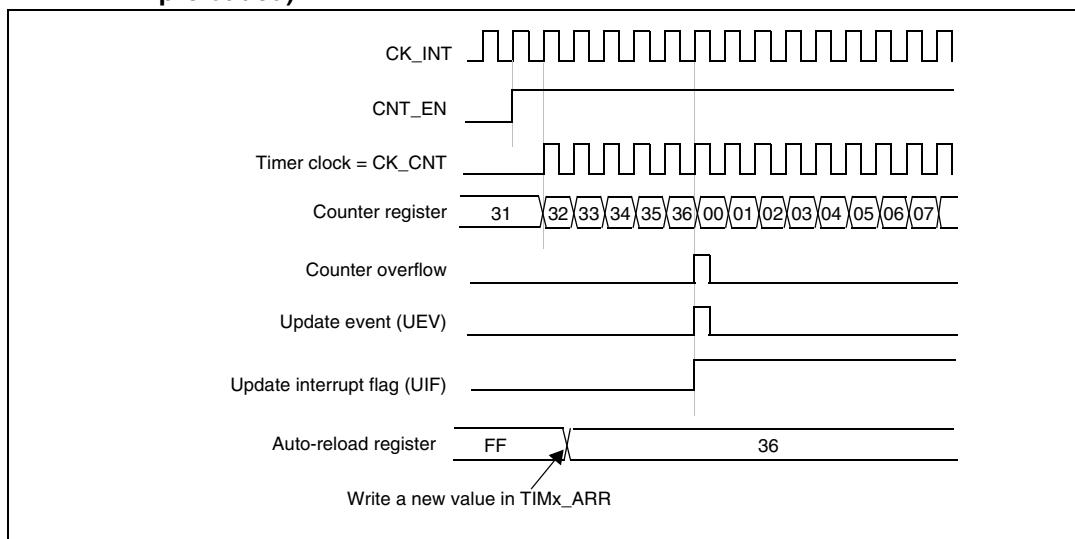
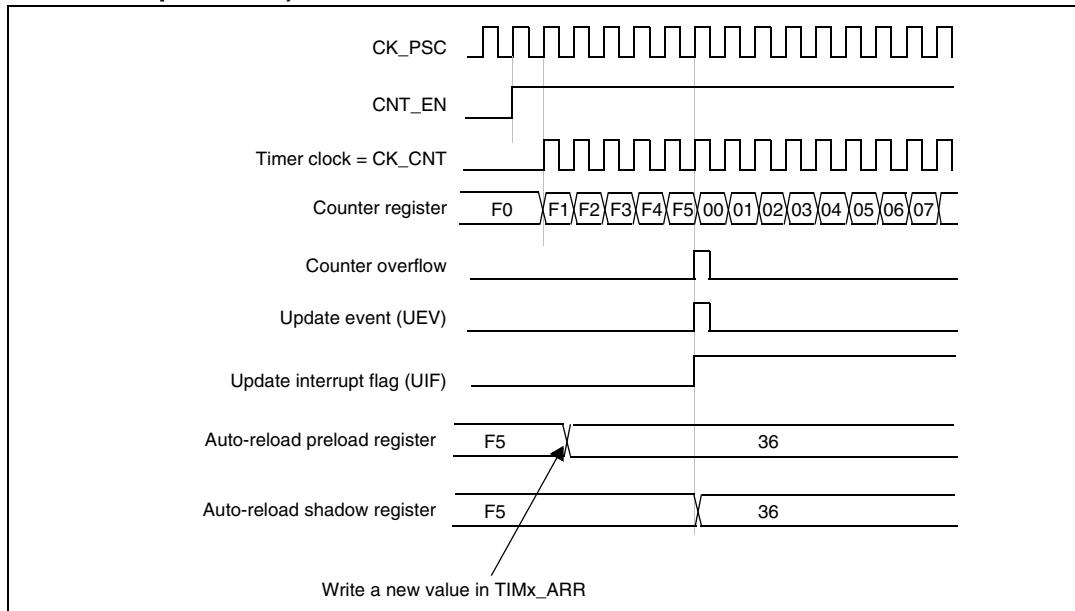
Figure 181. Counter timing diagram, internal clock divided by 4**Figure 182. Counter timing diagram, internal clock divided by N****Figure 183. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)**

Figure 184. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



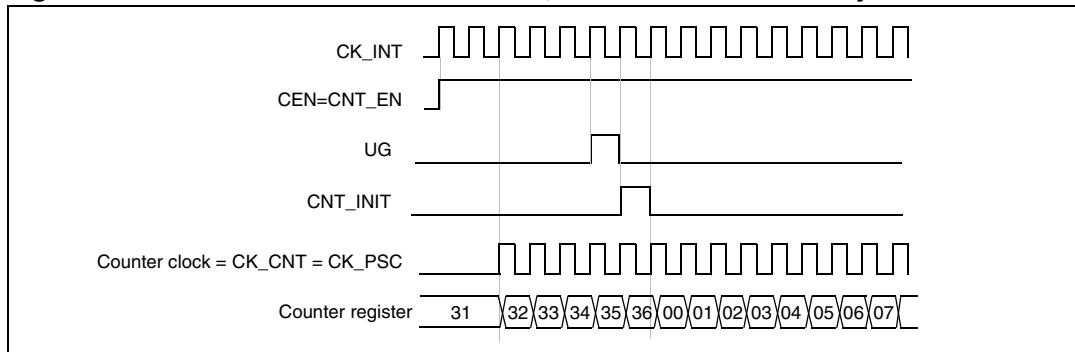
16.3.3 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 185 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 185. Control circuit in normal mode, internal clock divided by 1



16.3.4 Debug mode

When the microcontroller enters the debug mode (Cortex™-M4F 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 32.16.2: Debug support for timers, watchdog, bxCAN and I²C](#).

16.4 TIM6&TIM7 registers

Refer to [Section 1.1 on page 46](#) for a list of abbreviations used in register descriptions.

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

16.4.1 TIM6&TIM7 control register 1 (TIMx_CR1)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ARPE	Reserved		OPM	URS	UDIS	CEN	
								rw			rw	rw	rw	rw	

Bits 15:8 Reserved, 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.

16.4.2 TIM6&TIM7 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

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

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

Bits 6:4 **MMS**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).

010: **Update** - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

Bits 3:0 Reserved, must be kept at reset value.

16.4.3 TIM6&TIM7 DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

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

Bit 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable

- 0: Update DMA request disabled.
- 1: Update DMA request enabled.

Bit 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

- 0: Update interrupt disabled.
- 1: Update interrupt enabled.

16.4.4 TIM6&TIM7 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UIF rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

16.4.5 TIM6&TIM7 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UG w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

16.4.6 TIM6&TIM7 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]**: Counter value

16.4.7 TIM6&TIM7 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to $f_{CK_PSC} / (PSC[15:0] + 1)$.

PSC contains the value to be loaded into the active prescaler register at each update event.

16.4.8 TIM6&TIM7 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded into the actual auto-reload register.

Refer to [Section 16.3.1: Time-base unit on page 457](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

16.4.9 TIM6&TIM7 register map

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

Table 67. TIM6&TIM7 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	TIMx_CR1 Reset value																									ARPE 0	Reserved 0	OPM 0	URS 0	UDIS 0	UIE 0	CEN 0		
0x04	TIMx_CR2 Reset value																								MMS[2:0] 0 0 0	Reserved 0								
0x08																																		
0x0C	TIMx_DIER Reset value																								UDE 0	Reserved 0					UIE 0			
0x10	TIMx_SR Reset value																														UIF 0			
0x14	TIMx_EGR Reset value																														UG 0			
0x18																																		
0x1C																																		
0x20																																		
0x24	TIMx_CNT Reset value																								CNT[15:0]									
0x28	TIMx_PSC Reset value																								PSC[15:0]									
0x2C	TIMx_ARR Reset value																								ARR[15:0]									

Refer to [Table 1 on page 50](#) for the register boundary addresses.

17 Independent watchdog (IWDG)

17.1 IWDG introduction

The STM32F40x and STM32F41x have two embedded watchdog peripherals which offer a combination of high safety level, timing accuracy and flexibility of use. Both watchdog peripherals (Independent and Window) serve to detect and resolve malfunctions due to software failure, and to trigger system reset or an interrupt (window watchdog only) when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails. The window watchdog (WWDG) clock is prescaled from the APB1 clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. The WWDG is best suited to applications which require the watchdog to react within an accurate timing window. For further information on the window watchdog, refer to [Section 18 on page 472](#).

17.2 IWDG main features

- Free-running downcounter
- clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Reset (if watchdog activated) when the downcounter value of 0x000 is reached

17.3 IWDG functional description

[Figure 186](#) shows the functional blocks of the independent watchdog module.

When the independent watchdog is started by writing the value 0xCCCC in the Key register (IWDG_KR), the counter starts counting down from the reset value of 0xFFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0xAAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

17.3.1 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and will generate a reset unless the Key register is written by the software before the counter reaches end of count.

17.3.2 Register access protection

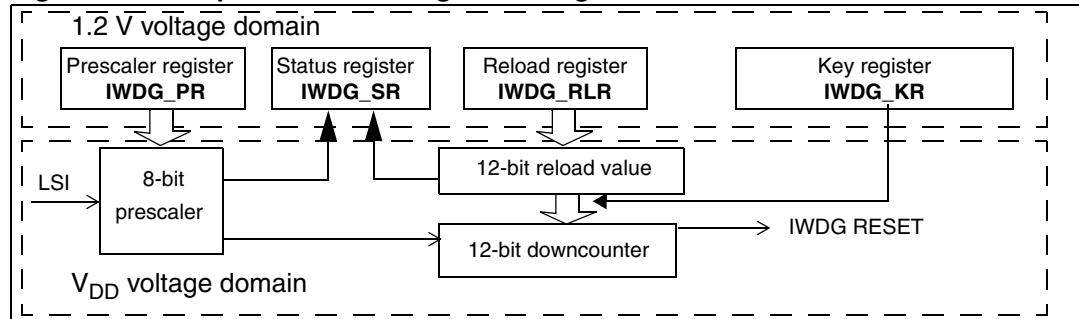
Write access to the IWDG_PR and IWDG_RLR registers is protected. To modify them, you must first write the code 0x5555 in the IWDG_KR register. A write access to this register with a different value will break the sequence and register access will be protected again. This implies that it is the case of the reload operation (writing 0xAAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value is on going.

17.3.3 Debug mode

When the microcontroller enters debug mode (Cortex™-M4F core halted), the IWDG counter either continues to work normally or stops, depending on DBG_IWDG_STOP configuration bit in DBG module. For more details, refer to [Section 32.16.2: Debug support for timers, watchdog, bxCAN and I²C](#).

Figure 186. Independent watchdog block diagram



Note:

The watchdog function is implemented in the V_{DD} voltage domain that is still functional in Stop and Standby modes.

Table 68. Min/max IWDG timeout period at 32 kHz (LSI)⁽¹⁾

Prescaler divider	PR[2:0] bits	Min timeout (ms) RL[11:0]=0x000	Max timeout (ms) RL[11:0]=0xFFFF
/4	0	0.125	512
/8	1	0.25	1024
/16	2	0.5	2048
/32	3	1	4096
/64	4	2	8192
/128	5	4	16384
/256	6	8	32768

- These timings are given for a 32 kHz clock but the microcontroller's internal RC frequency can vary from 30 to 60 kHz. Moreover, given an exact RC oscillator frequency, the exact timings still depend on the phasing of the APB interface clock versus the LSI clock so that there is always a full RC period of uncertainty.

17.4 IWDG registers

Refer to [Section 1.1 on page 46](#) for a list of abbreviations used in register descriptions.

17.4.1 Key register (IWDG_KR)

Address offset: 0x00

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

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

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

Bits 15:0 **KEY[15:0]**: Key value (write only, read 0000h)

These bits must be written by software at regular intervals with the key value AAAAh, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 5555h to enable access to the IWDG_PR and IWDG_RLR registers (see [Section 17.3.2](#))

Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

17.4.2 Prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

Bits 2:0 **PR[2:0]: Prescaler divider**

These bits are write access protected see [Section 17.3.2](#). They are written by software to select the prescaler divider feeding the counter clock. PVU bit of IWDG_SR must be reset in order to be able to change the prescaler divider.

- 000: divider /4
- 001: divider /8
- 010: divider /16
- 011: divider /32
- 100: divider /64
- 101: divider /128
- 110: divider /256
- 111: divider /256

Note: Reading this register returns the prescaler value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG_SR register is reset.

17.4.3 Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
Reserved																											RL[11:0]																			
																											rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	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 17.3.2](#). They are written by software to define the value to be loaded in the watchdog counter each time the value AAAAh is written in the IWDG_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to [Table 68](#).

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.

17.4.4 Status register (IWDG_SR)

Address offset: 0x0C

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

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

Reserved

RVU	PVU
r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RVU**: Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU**: Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Prescaler value can be updated only when PVU bit is reset.

Note: *If several reload values or prescaler values are used by application, it is mandatory to wait until RVU bit is reset before changing the reload value and to wait until PVU bit is reset before changing the prescaler value. However, after updating the prescaler and/or the reload value it is not necessary to wait until RVU or PVU is reset before continuing code execution (even in case of low-power mode entry, the write operation is taken into account and will complete)*

17.4.5 IWDG register map

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

Table 69. IWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Reserved												KEY[15:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x04	IWDG_PR	Reserved												PR[2:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x08	IWDG_RLR	Reserved												RL[11:0]																			
	Reset value													1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1				
0x0C	IWDG_SR	Reserved												RVU[PVU]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Refer to [Table 1 on page 50](#) for the register boundary addresses.

18 Window watchdog (WWDG)

18.1 WWDG introduction

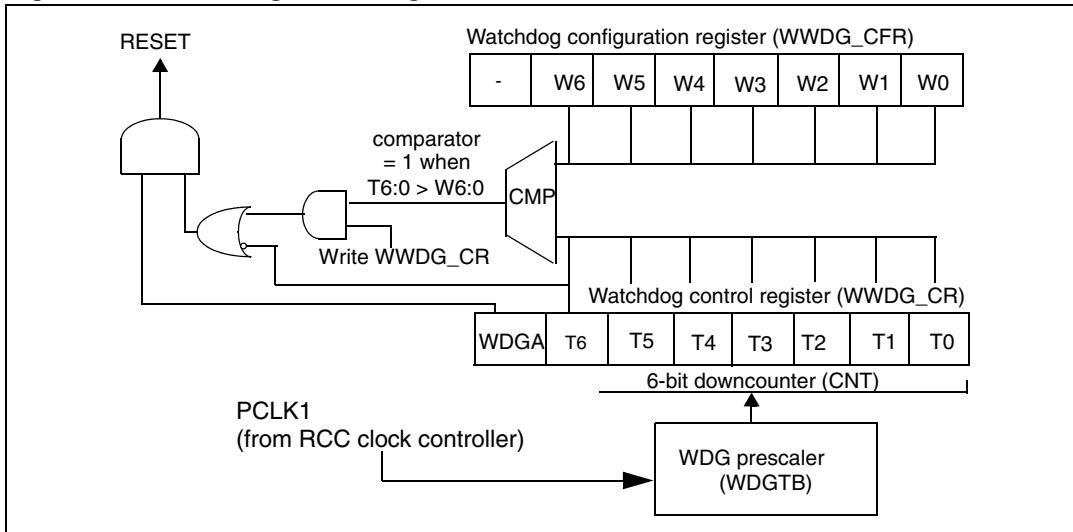
The window watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

18.2 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 188](#))
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

18.3 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 187. Watchdog block diagram

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0:

Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

Controlling the downcounter

This downcounter is free-running: It counts down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register (see [Figure 188](#)). 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 188](#) describes the window watchdog process.

Note: *The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).*

Advanced watchdog interrupt feature

The Early Wakeup Interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this

case, the corresponding interrupt service routine (ISR) should reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

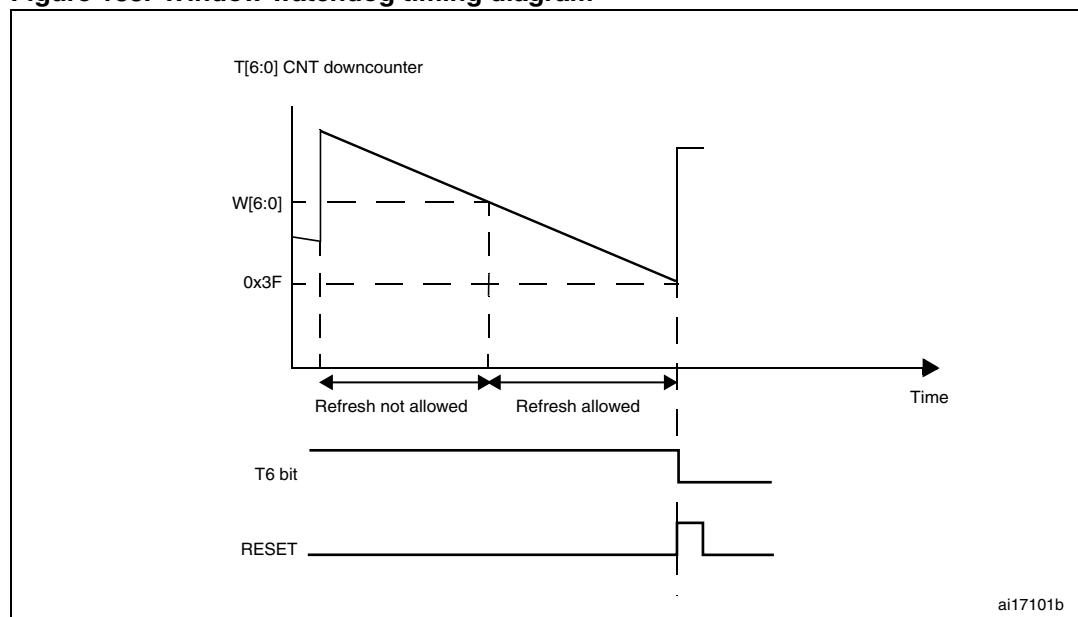
Note: *When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset will eventually be generated.*

18.4 How to program the watchdog timeout

You can use the formula in [Figure 188](#) 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 188. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$T_{\text{WWDG}} = T_{\text{PCLK1}} \times 4096 \times 2^{\text{WDGTB}} \times (T[5:0] + 1) \text{ (in ms)}$$

where

T_{WWDG} is the WWDG timeout

T_{PCLK1} is the APB1 clock period expressed in ms.

Refer to [Table 70](#) for the minimum and maximum values of the T_{WWDG} .

Table 70. Timeout values at 30 MHz (f_{PCLK1})

Prescaler	WDGTB	Min timeout (μ s) $T[5:0] = 0x00$	Max timeout (ms) $T[5:0] = 0x3F$
1	0	136.53	8.74
2	1	273.07	17.48
4	2	546.13	34.95
8	3	1092.27	69.91

18.5 Debug mode

When the microcontroller enters debug mode (Cortex™-M4F core halted), the WWDG counter either continues to work normally or stops, depending on DBG_WWDG_STOP configuration bit in DBG module. For more details, refer to [Section 32.16.2: Debug support for timers, watchdog, bxCAN and I²C](#).

18.6 WWDG registers

Refer to for a list of abbreviations used in register descriptions.

18.6.1 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	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								WDGA	T[6:0]							
								rs	rw							

Bits 31:8 Reserved, 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 $(4096 \times 2^{\text{WDGTB}})$ PCLK1 cycles. A reset is produced when it rolls over from 0x40 to 0x3F (T6 becomes cleared).

18.6.2 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		
Reserved																	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved								EWI	WDGTB[1:0]	W[6:0]							
								rs	rw	rw							

Bit 31:10 Reserved, 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 (PCLK1 div 4096) div 1

01: CK Counter Clock (PCLK1 div 4096) div 2

10: CK Counter Clock (PCLK1 div 4096) div 4

11: CK Counter Clock (PCLK1 div 4096) div 8

Bits 6:0 **W[6:0]**: 7-bit window value

These bits contain the window value to be compared to the downcounter.

18.6.3 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
															rc_w0

Bit 31:1 Reserved, 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.

18.6.4 WWDG register map

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

Table 71. WWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	WWDG_CR	Reserved																WDG_A	T[6:0]														
		Reset value																	0	1	1	1	1	1	1	1	1	0					
0x04	WWDG_CFR	Reserved																EWI	W[6:0]														
		Reset value																	0	0	0	1	1	1	1	1	1	1					
0x08	WWDG_SR	Reserved																EWIF															
		Reset value																	0	0	0	1	1	1	1	1	1	0					

Refer to [Table 1 on page 50](#) for the register boundary addresses.

19 Cryptographic processor (CRYP)

19.1 CRYP introduction

The cryptographic processor can be used to both encipher and decipher data using the DES, Triple-DES or AES (128, 192, or 256) algorithms. It is a fully compliant implementation of the following standards:

- The data encryption standard (DES) and Triple-DES (TDES) as defined by Federal Information Processing Standards Publication (FIPS PUB 46-3, 1999 October 25). It follows the American National Standards Institute (ANSI) X9.52 standard.
- The advanced encryption standard (AES) as defined by Federal Information Processing Standards Publication (FIPS PUB 197, 2001 November 26)

The CRYP processor performs data encryption and decryption using DES and TDES algorithms in Electronic codebook (ECB) or Cipher block chaining (CBC) mode.

The CRYP peripheral is a 32-bit AHB2 peripheral. It supports DMA transfer for incoming and processed data, and has input and output FIFOs (each 8 words deep).

19.2 CRYP main features

- Suitable for AES, DES and TDES enciphering and deciphering operations
- AES
 - Supports the ECB, CBC and CTR chaining algorithms
 - Supports 128-, 192- and 256-bit keys
 - 4 × 32-bit initialization vectors (IV) used in the CBC and CTR modes
 - 14 HCLK cycles to process one 128-bit block in AES
 - 16 HCLK cycles to process one 192-bit block in AES
 - 18 HCLK cycles to process one 256-bit block in AES
- DES/TDES
 - Direct implementation of simple DES algorithms (a single key, K1, is used)
 - Supports the ECB and CBC chaining algorithms
 - Supports 64-, 128- and 192-bit keys (including parity)
 - 2 × 32-bit initialization vectors (IV) used in the CBC mode
 - 16 HCLK cycles to process one 64-bit block in DES
 - 48 HCLK cycles to process one 64-bit block in TDES
- Common to DES/TDES and AES
 - IN and OUT FIFO (each with an 8-word depth, a 32-bit width, corresponding to 4 DES blocks or 2 AES blocks)
 - Automatic data flow control with support of direct memory access (DMA) (using 2 channels, one for incoming data the other for processed data)
 - Data swapping logic to support 1-, 8-, 16- or 32-bit data

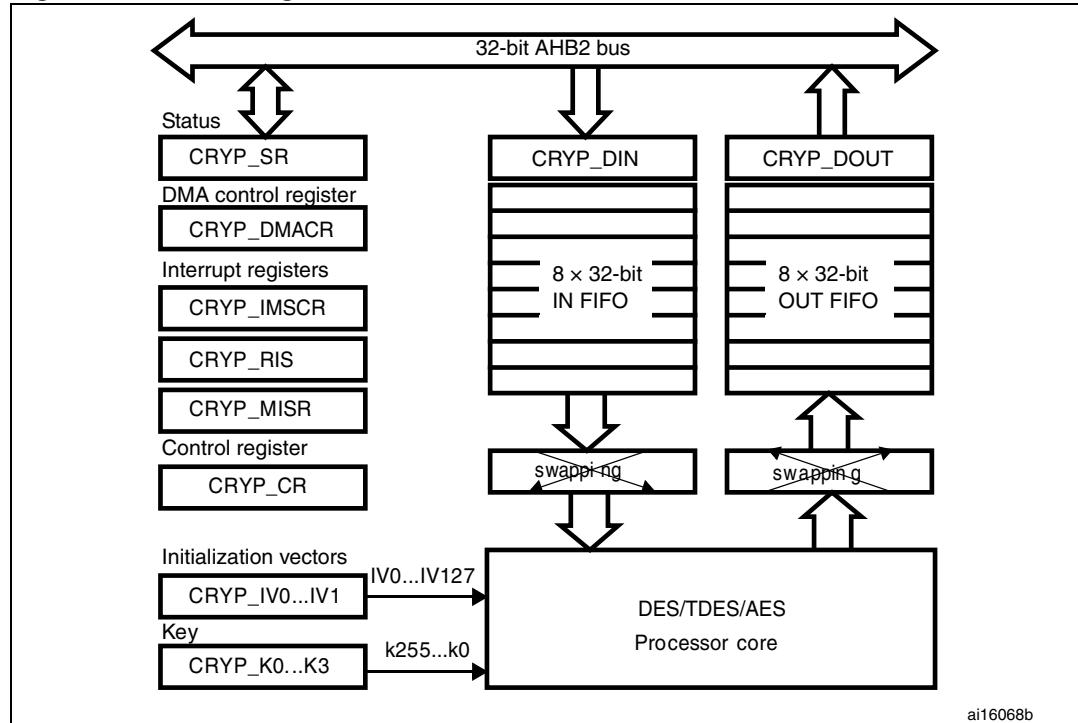
19.3 CRYP functional description

The cryptographic processor implements a Triple-DES (TDES, that also supports DES) core and an AES cryptographic core. [Section 19.3.1](#) and [Section 19.3.2](#) provide details on these cores.

Since the TDES and the AES algorithms use block ciphers, incomplete input data blocks have to be padded prior to encryption (extra bits should be appended to the trailing end of the data string). After decryption, the padding has to be discarded. The hardware does not manage the padding operation, the software has to handle it.

[Figure 189](#) shows the block diagram of the cryptographic processor.

Figure 189. Block diagram



19.3.1 DES/TDES cryptographic core

The DES/TDES cryptographic core consists of three components:

- The DES algorithm (DEA)
- Multiple keys (1 for the DES algorithm, 1 to 3 for the TDES algorithm)
- The initialization vector (used in the CBC mode)

The basic processing involved in the TDES is as follows: an input block is read in the DEA and encrypted using the first key, K1 (K0 is not used in TDES mode). The output is then decrypted using the second key, K2, and encrypted using the third key, K3. The key depends on the algorithm which is used:

- DES mode: Key = [K1]
- TDES mode: Key = [K3 K2 K1]

where $Kx = [KxR \ KxL]$, R = right, L = left

According to the mode implemented, the resultant output block is used to calculate the ciphertext.

Note that the outputs of the intermediate DEA stages is never revealed outside the cryptographic boundary.

The TDES allows three different keying options:

- Three independent keys

The first option specifies that all the keys are independent, that is, K1, K2 and K3 are independent. FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to this option as the Keying Option 1 and, to the TDES as 3-key TDES.

- Two independent keys

The second option specifies that K1 and K2 are independent and K3 is equal to K1, that is, K1 and K2 are independent, $K3 = K1$. FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to this second option as the Keying Option 2 and, to the TDES as 2-key TDES.

- Three equal keys

The third option specifies that K1, K2 and K3 are equal, that is, $K1 = K2 = K3$. FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to the third option as the Keying Option 3. This “1-key” TDES is equivalent to single DES.

FIPS PUB 46-3 – 1999 (and ANSI X9.52-1998) provides a thorough explanation of the processing involved in the four operation modes supplied by the TDEA (TDES algorithm): TDES-ECB encryption, TDES-ECB decryption, TDES-CBC encryption and TDES-CBC decryption.

This reference manual only gives a brief explanation of each mode.

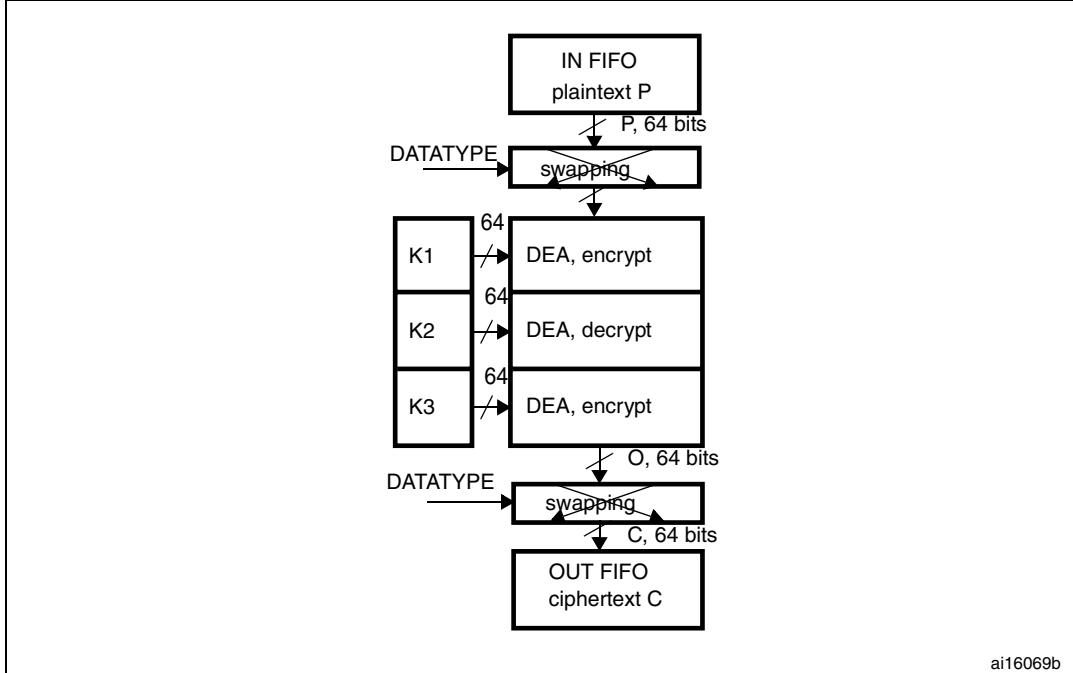
DES and TDES Electronic codebook (DES/TDES-ECB) mode

- DES/TDES-ECB mode encryption

Figure 190 illustrates the encryption in DES and TDES Electronic codebook (DES/TDES-ECB) mode. A 64-bit plaintext data block (P) is used after bit/byte/half-word swapping (refer to [Section 19.3.3: Data type on page 492](#)) as the input block (I). The input block is processed through the DEA in the encrypt state using K1. The output of this process is fed back directly to the input of the DEA where the DES is performed in the decrypt state using K2. The output of this process is fed back directly to the input of the DEA where the DES is performed in the encrypt state using K3. The resultant 64-bit output block (O) is used, after bit/byte/half-word swapping, as ciphertext (C) and it is pushed into the OUT FIFO.

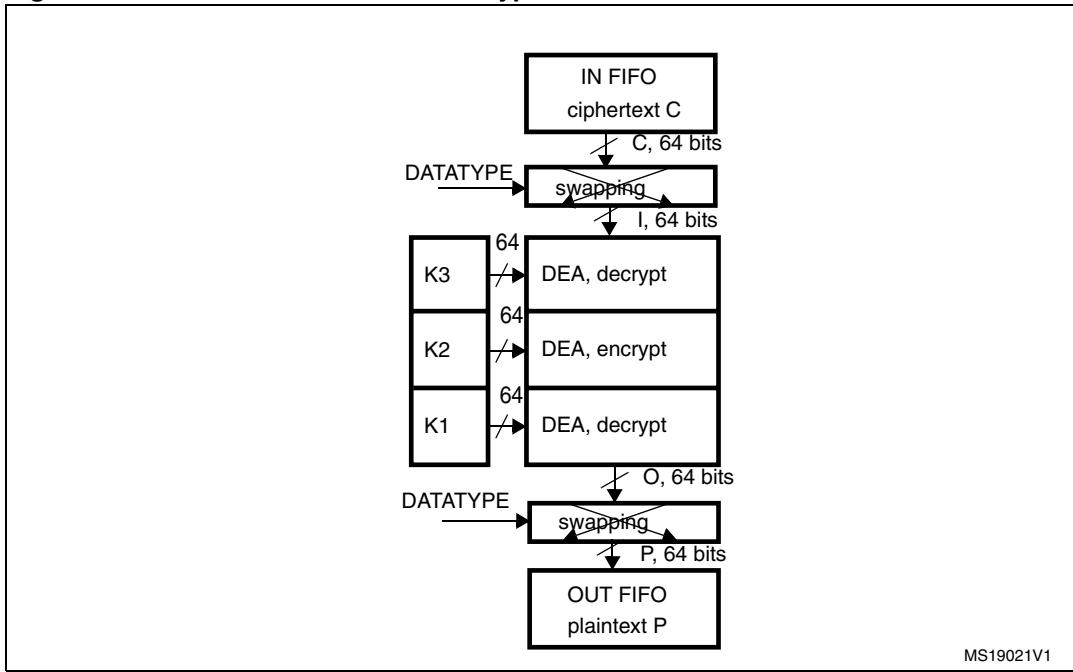
- DES/TDES-ECB mode decryption

Figure 191 illustrates the DES/TDES-ECB decryption. A 64-bit ciphertext block (C) is used, after bit/byte/half-word swapping, as the input block (I). The keying sequence is reversed compared to that used in the encryption process. The input block is processed through the DEA in the decrypt state using K3. The output of this process is fed back directly to the input of the DEA where the DES is performed in the encrypt state using K2. The new result is directly fed to the input of the DEA where the DES is performed in the decrypt state using K1. The resultant 64-bit output block (O), after bit/byte/half-word swapping, produces the plaintext (P).

Figure 190. DES/TDES-ECB mode encryption

ai16069b

1. K: key; C: cipher text; I: input block; O: output block; P: plain text.

Figure 191. DES/TDES-ECB mode decryption

MS19021V1

1. K: key; C: cipher text; I: input block; O: output block; P: plain text.

DES and TDES Cipher block chaining (DES/TDES-CBC) mode

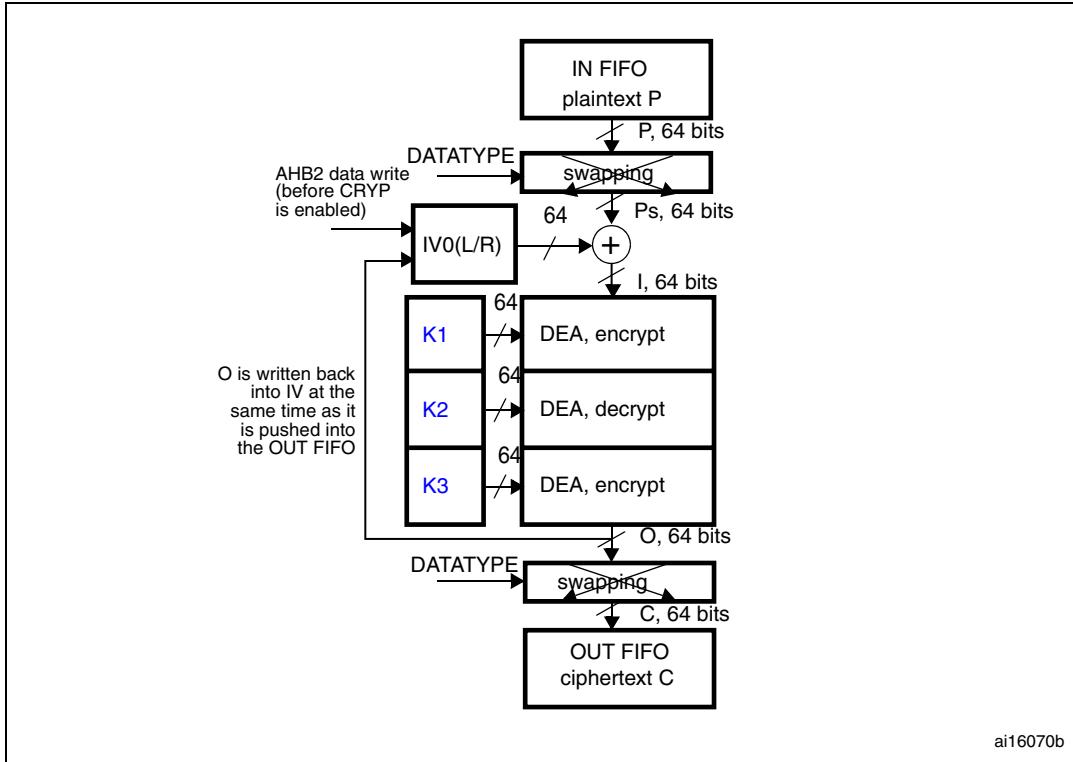
- DES/TDES-CBC mode encryption

Figure 192 illustrates the DES and Triple-DES Cipher block chaining (DES/TDES-CBC) mode encryption. This mode begins by dividing a plaintext message into 64-bit data blocks. In TCBC encryption, the first input block (I_1), obtained after bit/byte/half-word swapping (refer to [Section 19.3.3: Data type on page 492](#)), is formed by exclusive-ORing the first plaintext data block (P_1) with a 64-bit initialization vector IV ($I_1 = IV \oplus P_1$). The input block is processed through the DEA in the encrypt state using K1. The output of this process is fed back directly to the input of the DEA, which performs the DES in the decrypt state using K2. The output of this process is fed directly to the input of the DEA, which performs the DES in the encrypt state using K3. The resultant 64-bit output block (O_1) is used directly as the ciphertext (C_1), that is, $C_1 = O_1$. This first ciphertext block is then exclusive-ORed with the second plaintext data block to produce the second input block, ($I_2 = C_1 \oplus P_2$). Note that I_2 and P_2 now refer to the second block. The second input block is processed through the TDEA to produce the second ciphertext block. This encryption process continues to “chain” successive cipher and plaintext blocks together until the last plaintext block in the message is encrypted. If the message does not consist of an integral number of data blocks, then the final partial data block should be encrypted in a manner specified for the application.

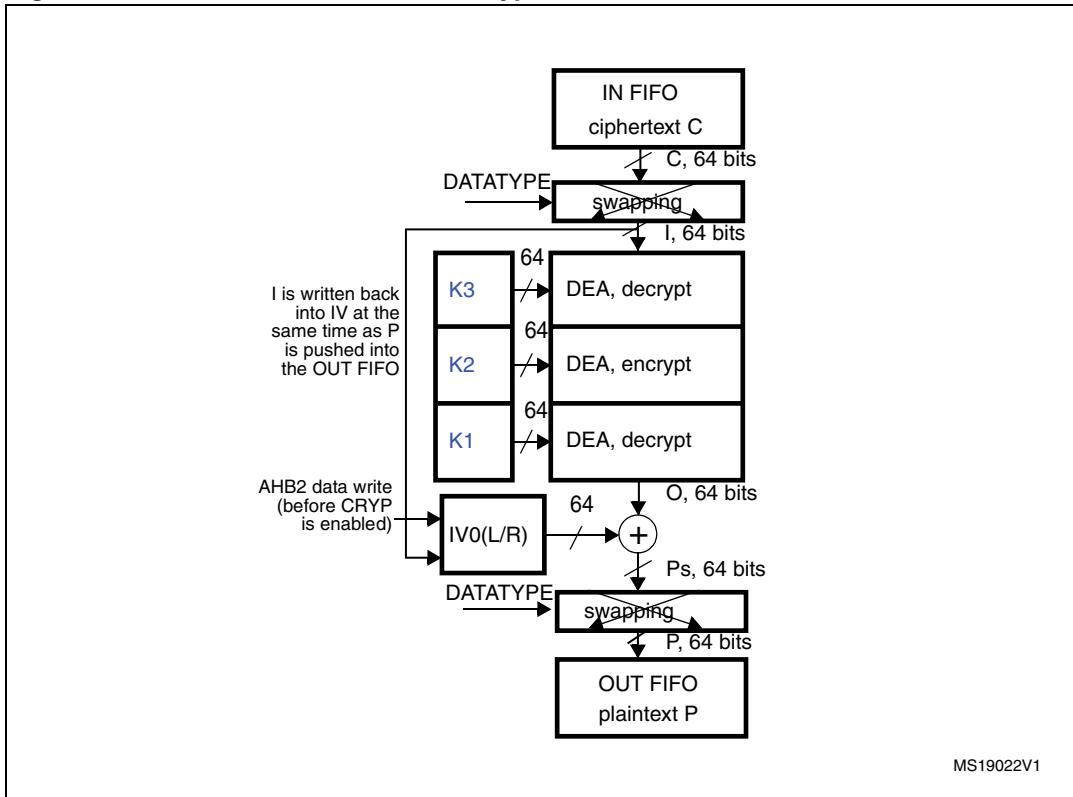
- DES/TDES-CBC mode decryption

In DES/TDES-CBC decryption (see *Figure 193*), the first ciphertext block (C_1) is used directly as the input block (I_1). The keying sequence is reversed compared to that used for the encrypt process. The input block is processed through the DEA in the decrypt state using K3. The output of this process is fed directly to the input of the DEA where the DES is processed in the encrypt state using K2. This resulting value is directly fed to the input of the DEA where the DES is processed in the decrypt state using K1. The resulting output block is exclusive-ORed with the IV (which must be the same as that used during encryption) to produce the first plaintext block ($P_1 = O_1 \oplus IV$). The second ciphertext block is then used as the next input block and is processed through the TDEA. The resulting output block is exclusive-ORed with the first ciphertext block to produce the second plaintext data block ($P_2 = O_2 \oplus C_1$). (Note that P_2 and O_2 refer to the second block of data.) The TCBC decryption process continues in this manner until the last complete ciphertext block has been decrypted. Ciphertext representing a partial data block must be decrypted in a manner specified for the application.

Figure 192. DES/TDES-CBC mode encryption



1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: initialization vectors.

Figure 193. DES/TDES-CBC mode decryption

1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: initialization vectors.

19.3.2 AES cryptographic core

The AES cryptographic core consists of three components:

- The AES algorithm (AEA: advanced encryption algorithm)
- Multiple keys
- Initialization vector(s)

The AES utilizes keys of 3 possible lengths: 128, 192 or 256 bits and, depending on the operation mode used, zero or one 128-bit initialization vector (IV).

The basic processing involved in the AES is as follows: an input block of 128 bits is read from the input FIFO and sent to the AEA to be encrypted using the key (K0...3). The key format depends on the key size:

- If Key size = 128: Key = [K3 K2]
- If Key size = 192: Key = [K3 K2 K1]
- If Key size = 256: Key = [K3 K2 K1 K0]

where $Kx=[KxR \ KxL]$, R=right, L=left

According to the mode implemented, the resultant output block is used to calculate the ciphertext.

FIPS PUB 197 (November 26, 2001) provides a thorough explanation of the processing involved in the four operation modes supplied by the AES core: AES-ECB encryption, AES-

ECB decryption, AES-CBC encryption and AES-CBC decryption. This reference manual only gives a brief explanation of each mode.

AES Electronic codebook (AES-ECB) mode

- AES-ECB mode encryption

Figure 194 illustrates the AES Electronic codebook (AES-ECB) mode encryption.

In AES-ECB encryption, a 128-bit plaintext data block (P) is used after bit/byte/half-word swapping (refer to [Section 19.3.3: Data type on page 492](#)) as the input block (I). The input block is processed through the AEA in the encrypt state using the 128, 192 or 256-bit key. The resultant 128-bit output block (O) is used after bit/byte/half-word swapping as ciphertext (C). It is then pushed into the OUT FIFO.

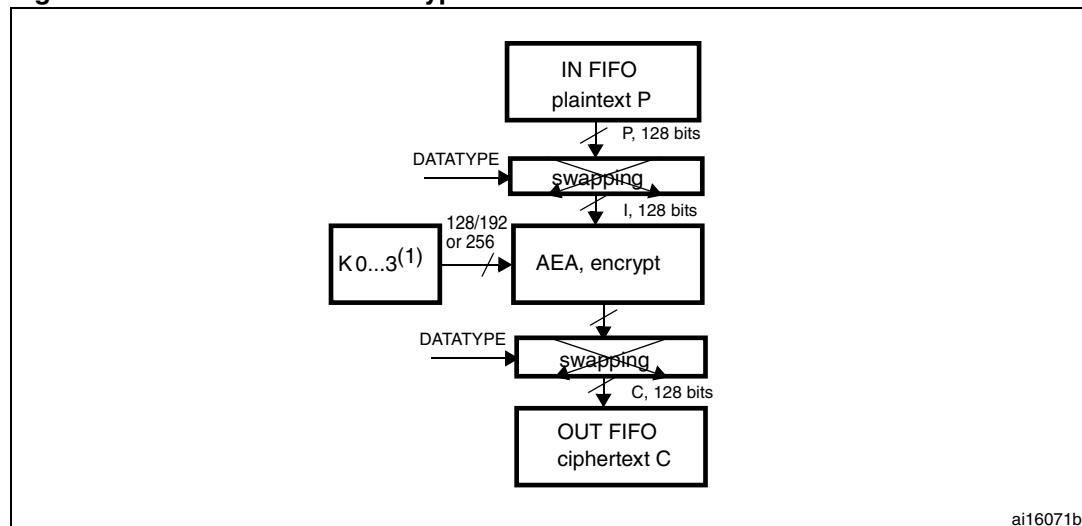
- AES-ECB mode decryption

Figure 195 illustrates the AES Electronic codebook (AES-ECB) mode encryption.

To perform an AES decryption in the ECB mode, the secret key has to be prepared (it is necessary to execute the complete key schedule for encryption) by collecting the last round key, and using it as the first round key for the decryption of the ciphertext. This preparation function is computed by the AES core. Refer to [Section 19.3.6: Procedure to perform an encryption or a decryption](#) for more details on how to prepare the key.

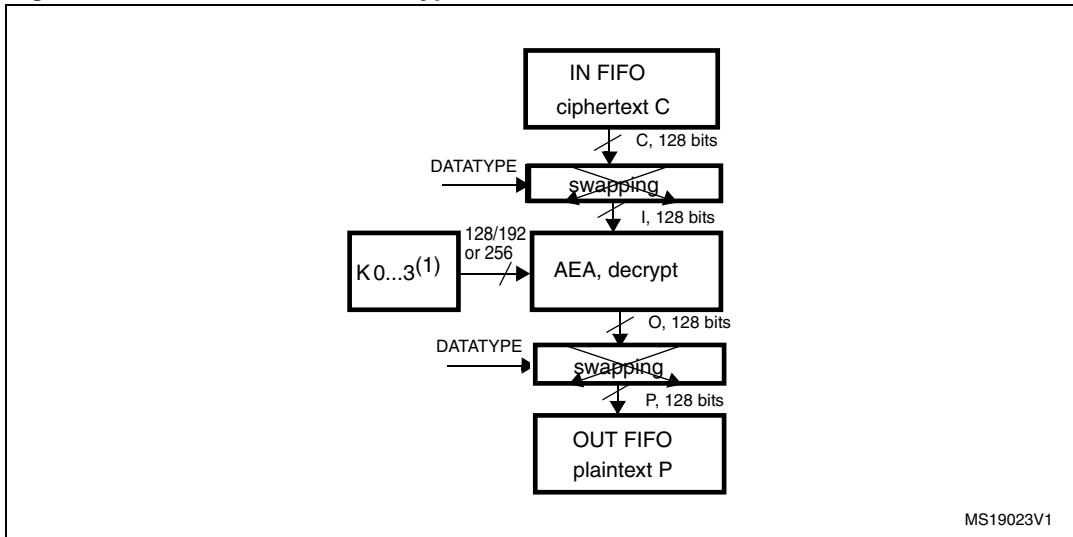
In AES-ECB decryption, a 128-bit ciphertext block (C) is used after bit/byte/half-word swapping as the input block (I). The keying sequence is reversed compared to that of the encryption process. The resultant 128-bit output block (O), after bit/byte or half-word swapping, produces the plaintext (P).

Figure 194. AES-ECB mode encryption



ai16071b

1. K: key; C: cipher text; I: input block; O: output block; P: plain text.
2. If Key size = 128: Key = [K3 K2].
If Key size = 192: Key = [K3 K2 K1]
If Key size = 256: Key = [K3 K2 K1 K0].

Figure 195. AES-ECB mode decryption

1. K: key; C: cipher text; I: input block; O: output block; P: plain text.
2. If Key size = 128 => Key = [K3 K2].
If Key size = 192 => Key = [K3 K2 K1]
If Key size = 256 => Key = [K3 K2 K1 K0].

AES Cipher block chaining (AES-CBC) mode

- AES-CBC mode encryption

The AES Cipher block chaining (AES-CBC) mode decryption is shown on [Figure 196](#).

In AES-CBC encryption, the first input block (I_1) obtained after bit/byte/half-word swapping (refer to [Section 19.3.3: Data type on page 492](#)) is formed by exclusive-ORing the first plaintext data block (P_1) with a 128-bit initialization vector IV ($I_1 = IV \oplus P_1$). The input block is processed through the AEA in the encrypt state using the 128-, 192- or 256-bit key ($K_0 \dots K_3$). The resultant 128-bit output block (O_1) is used directly as ciphertext (C_1), that is, $C_1 = O_1$. This first ciphertext block is then exclusive-ORed with the second plaintext data block to produce the second input block, ($I_2 = (C_1 \oplus P_2)$). Note that I_2 and P_2 now refer to the second block. The second input block is processed through the AEA to produce the second ciphertext block. This encryption process continues to “chain” successive cipher and plaintext blocks together until the last plaintext block in the message is encrypted. If the message does not consist of an integral number of data blocks, then the final partial data block should be encrypted in a manner specified for the application.

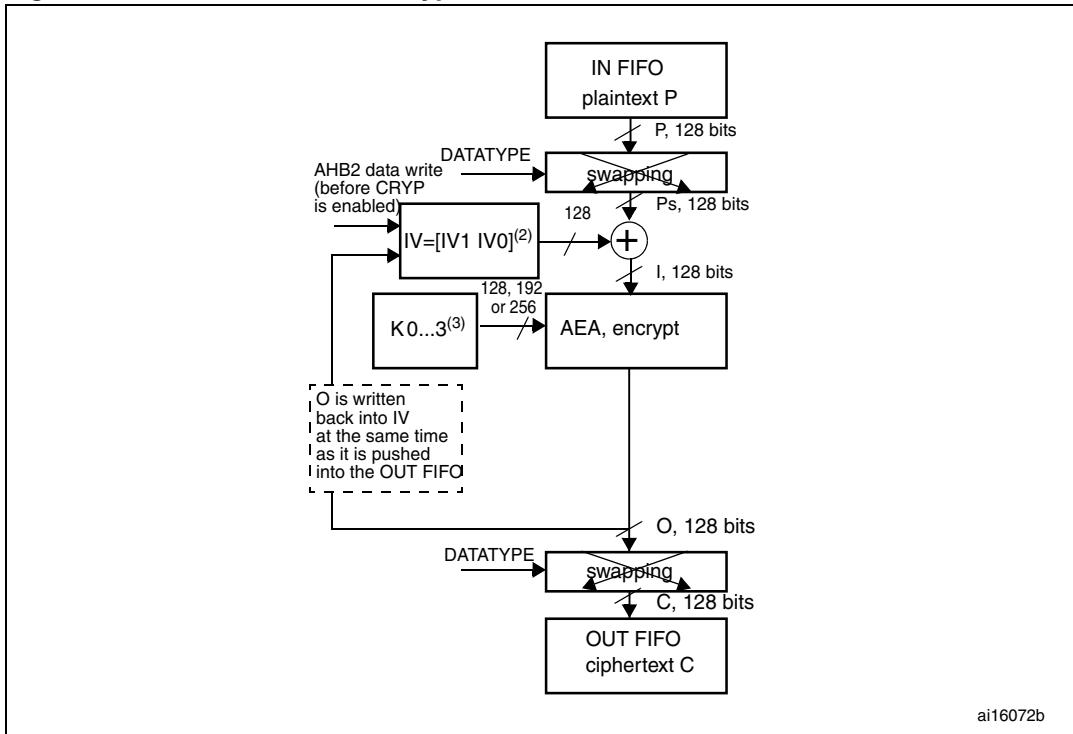
In the CBC mode, like in the ECB mode, the secret key must be prepared to perform an AES decryption. Refer to [Section 19.3.6: Procedure to perform an encryption or a decryption on page 497](#) for more details on how to prepare the key.

- AES-CBC mode decryption

In AES-CBC decryption (see [Figure 197](#)), the first 128-bit ciphertext block (C_1) is used directly as the input block (I_1). The input block is processed through the AEA in the decrypt state using the 128-, 192- or 256-bit key. The resulting output block is exclusive-ORed with the 128-bit initialization vector IV (which must be the same as that used during encryption) to produce the first plaintext block ($P_1 = O_1 \oplus IV$). The second ciphertext block is then used as the next input block and is processed through the AEA. The resulting output block is exclusive-ORed with the first ciphertext block to produce the second plaintext data block ($P_2 = O_2 \oplus C_1$). (Note that P_2 and O_2 refer to the second

block of data.) The AES-CBC decryption process continues in this manner until the last complete ciphertext block has been decrypted. Ciphertext representing a partial data block must be decrypted in a manner specified for the application.

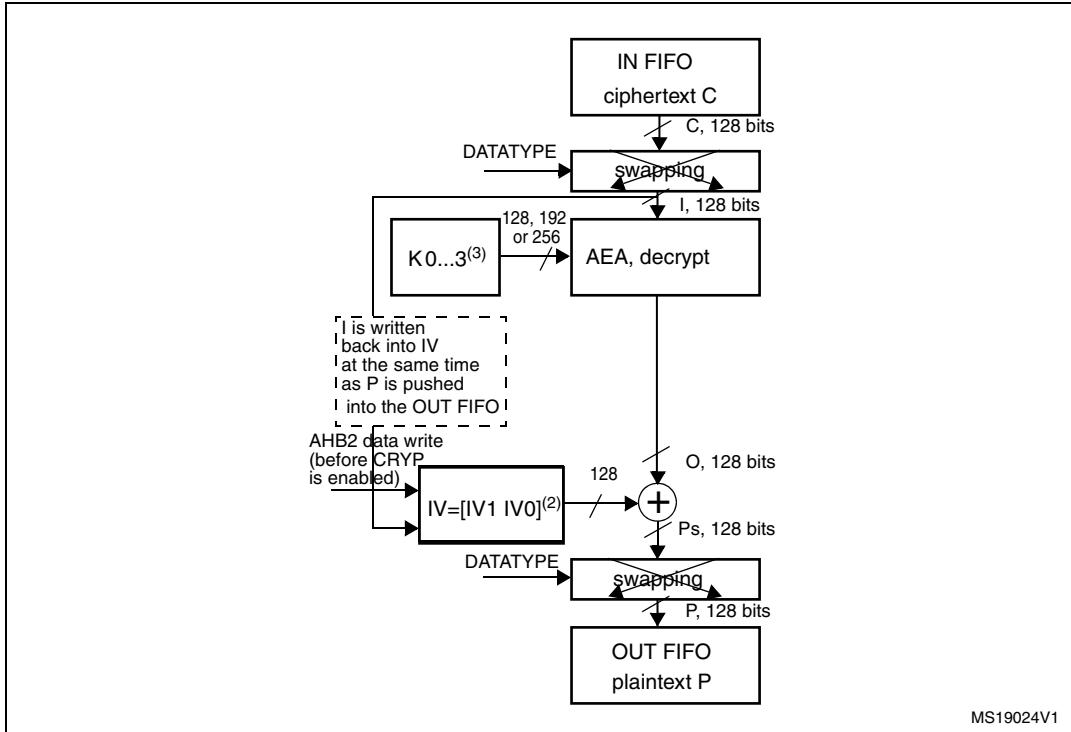
Figure 196. AES-CBC mode encryption



ai16072b

1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: Initialization vectors.
2. IVx=[IVxR IVxL], R=right, L=left.
3. If Key size = 128 => Key = [K3 K2].
If Key size = 192 => Key = [K3 K2 K1].
If Key size = 256 => Key = [K3 K2 K1 K0].

Figure 197. AES-CBC mode decryption



MS19024V1

1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: Initialization vectors.
2. $IVx = [IVxR \ IVxL]$, R=right, L=left.
3. If Key size = 128 => Key = [K3 K2].
If Key size = 192 => Key = [K3 K2 K1]
If Key size = 256 => Key = [K3 K2 K1 K0].

AES counter mode (AES-CTR) mode

The AES Counter mode uses the AES block as a key stream generator. The generated keys are then XORed with the plaintext to obtain the cipher. For this reason, it makes no sense to speak of different CTR encryption/decryption, since the two operations are exactly the same.

In fact, given:

- Plaintext: $P[0], P[1], \dots, P[n]$ (128 bits each)
- A key K to be used (the size does not matter)
- An initial counter block (call it ICB but it has the same functionality as the IV of CBC)

The cipher is computed as follows:

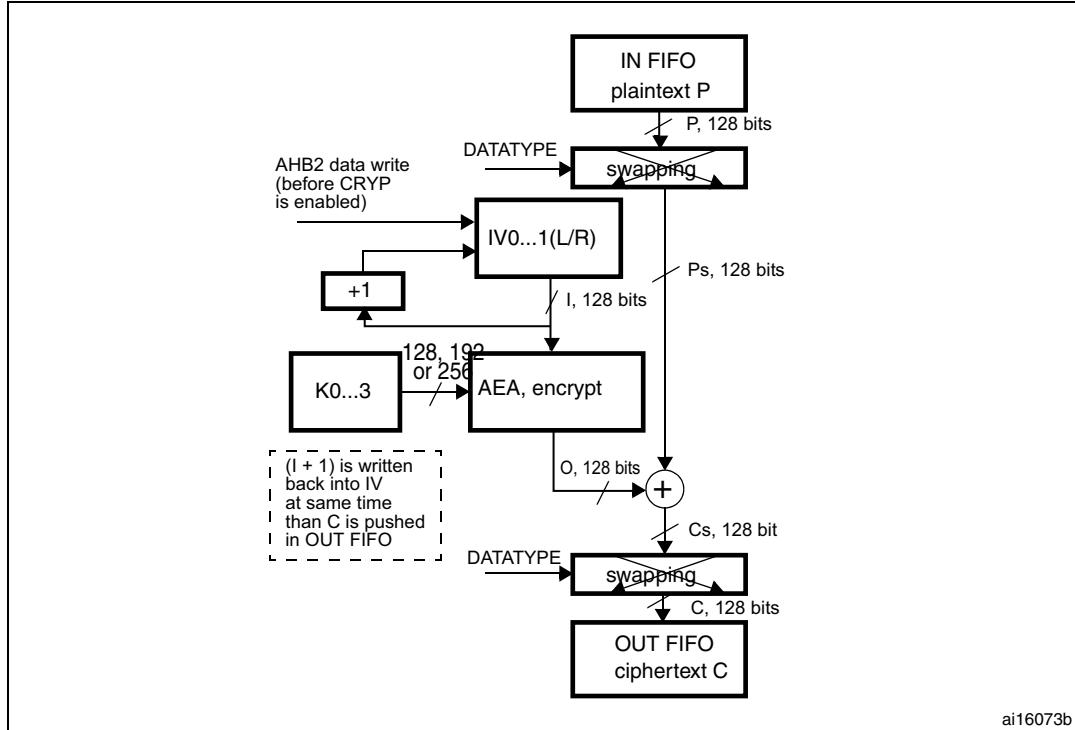
$C[i] = \text{enck}(iv[i]) \ xor \ P[i]$, where:

$iv[0] = \text{ICB}$ and $iv[i+1] = \text{func}(iv[i])$, where func is an update function applied to the previous iv block; func is basically an increment of one of the fields composing the iv block.

Given that the ICB for decryption is the same as the one for encryption, the key stream generated during decryption is the same as the one generated during encryption. Then, the ciphertext is XORed with the key stream in order to retrieve the original plaintext. The decryption operation therefore acts exactly in the same way as the encryption operation.

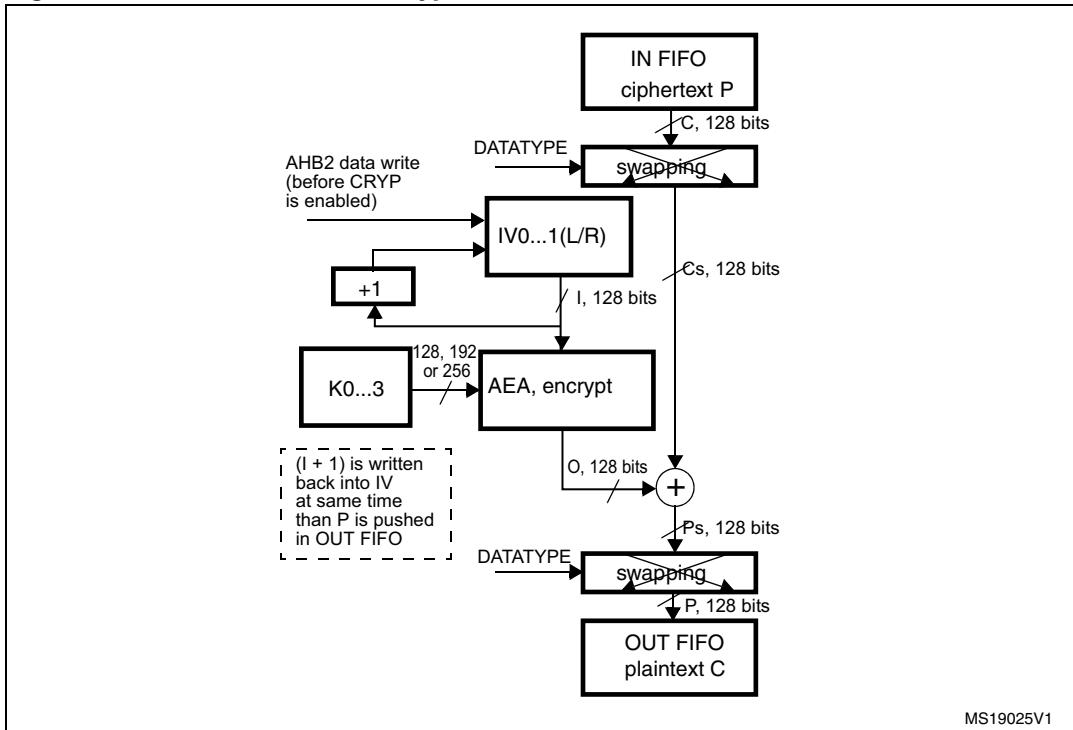
Figure 198 and *Figure 199* illustrate AES-CTR encryption and decryption, respectively.

Figure 198. AES-CTR mode encryption



ai16073b

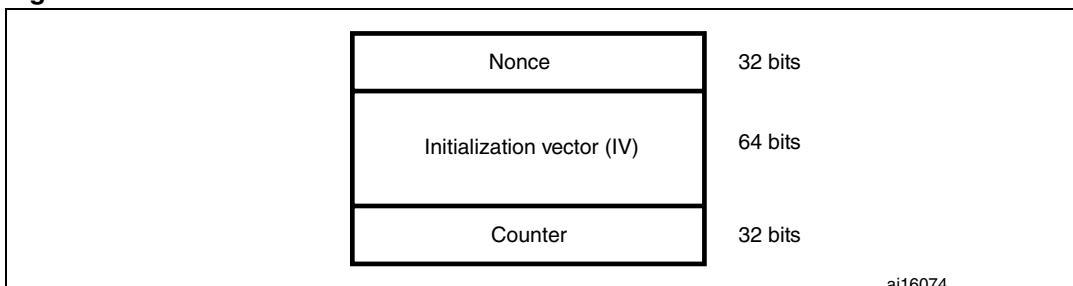
1. K: key; C: cipher text; I: input Block; o: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); Cs: cipher text after swapping (when decoding) or before swapping (when encoding); P: plain text; IV: Initialization vectors.

Figure 199. AES-CTR mode encryption

MS19025V1

1. K: key; C: cipher text; I: input Block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); Cs: cipher text after swapping (when decoding) or before swapping (when encoding); P: plain text; IV: Initialization vectors.

Figure 200 shows the structure of the IV block as defined by the standard [2]. It is composed of three distinct fields.

Figure 200. Initial counter block structure for the Counter mode

ai16074

- Nonce is a 32-bit, single-use value. A new nonce should be assigned to each different communication.
- The initialization vector (IV) is a 64-bit value and the standard specifies that the encryptor must choose IV so as to ensure that a given value is used only once for a given key
- The counter is a 32-bit big-endian integer that is incremented each time a block has been encrypted. The initial value of the counter should be set to 1.

The block increments the least significant 32 bits, while it leaves the other (most significant) 96 bits unchanged.

19.3.3 Data type

Data enter the CRYP processor 32 bits (word) at a time as they are written into the CRYP_DIN register. The principle of the DES is that streams of data are processed 64 bits by 64 bits and, for each 64-bit block, the bits are numbered from M1 to M64, with M1 the left-most bit and M64 the right-most bit of the block. The same principle is used for the AES, but with a 128-bit block size.

The system memory organization is little-endian: whatever the data type (bit, byte, 16-bit half-word, 32-bit word) used, the least-significant data occupy the lowest address locations. A bit, byte, or half-word swapping operation (depending on the kind of data to be encrypted) therefore has to be performed on the data read from the IN FIFO before they enter the CRYP processor. The same swapping operation should be performed on the CRYP data before they are written into the OUT FIFO. For example, the operation would be byte swapping for an ASCII text stream.

The kind of data to be processed is configured with the DATATYPE bitfield in the CRYP control register (CRYP_CR).

Table 72. Data types

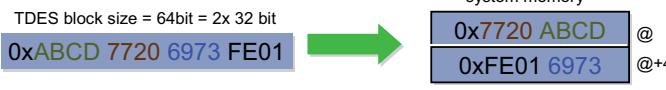
DATATYPE in CRYP_CR	Swapping performed	System memory data (plaintext or cipher)
00b	No swapping	<p>Example: TDES block value 0xABCD77206973FE01 is represented in system memory as:</p> <p>TDES block size = 64bit = 2x 32 bit</p> 
01b	Half-word (16-bit) swapping	<p>Example: TDES block value 0xABCD77206973FE01 is represented in system memory as:</p> <p>TDES block size = 64bit = 2x 32 bit</p> 

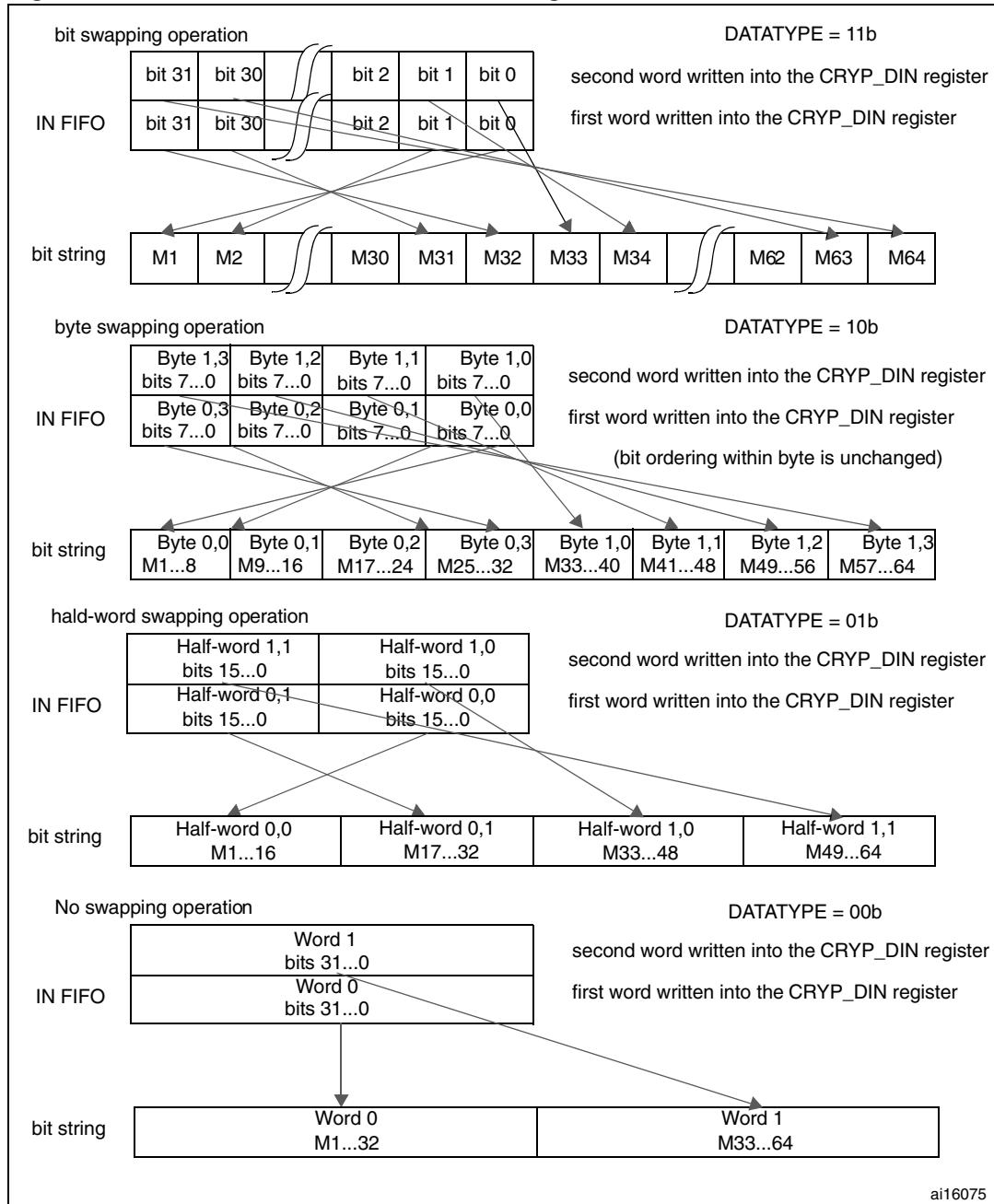
Table 72. Data types

DATATYPE in CRYP_CR	Swapping performed	System memory data (plaintext or cipher)
10b	Byte (8-bit) swapping	<p>Example: TDES block value 0xABCD77206973FE01 is represented in system memory as:</p> <p>TDES block size = 64bit = 2x 32 bit</p> <p>system memory 0x 20 77 CD AB @ 0x 01 FE 73 69 @+4</p>
11b	Bit swapping	<p>TDES block value 0x4E6F772069732074 is represented in system memory as:</p> <p>TDES Bloc size = 64bit = 2x 32 bit</p> <p>system memory 0x04 EE F6 72 @ 0x2E 04 CE 96 @+4</p> <p>0100 1110 0110 1111 0111 0111 0000 0010 0000 0111 0110 1111 0111 0010 0010 1110 0000 0100 1100 1110 1001 0110 1110 1111 0110 0111 0010 0010 1110 0000 0100 1100 1110 1001 0110 @+4</p>

Figure 201 shows how the 64-bit data block M1...64 is constructed from two consecutive 32-bit words popped off the IN FIFO by the CRYP processor, according to the DATATYPE value. The same schematic can easily be extended to form the 128-bit block for the AES cryptographic algorithm (for the AES, the block length is four 32-bit words, but swapping only takes place at word level, so it is identical to the one described here for the TDES).

Note: The same swapping is performed between the IN FIFO and the CRYP data block, and between the CRYP data block and the OUT FIFO.

Figure 201. 64-bit block construction according to DATATYPE



19.3.4 Initialization vectors - CRYP_IV0...1(L/R)

Initialization vectors are considered as two 64-bit data items. They therefore do not have the same data format and representation in system memory as plaintext or cypher data, and they are not affected by the DATATYPE value.

Initialization vectors are defined by two consecutive 32-bit words, CRYP_IVL (left part, noted as bits IV1...32) and CRYP_IVR (right part, noted as bits IV33...64).

During the DES or TDES CBC encryption, the CRYP_IV0(L/R) bits are XORed with the 64-bit data block popped off the IN FIFO after swapping (according to the DATATYPE value), that is, with the M1...64 bits of the data block. When the output of the [DEA3](#) block is available, it is copied back into the CRYP_IV0(L/R) vector, and this new content is XORed with the next 64-bit data block popped off the IN FIFO, and so on.

During the DES or TDES CBC decryption, the CRYP_IV0(L/R) bits are XORed with the 64-bit data block (that is, with the M1...64 bits) delivered by the [TDEA1](#) block before swapping (according to the DATATYPE value), and pushed into the OUT FIFO. When the XORed result is swapped and pushed into the OUT FIFO, the CRYP_IV0(L/R) value is replaced by the output of the IN FIFO, then the IN FIFO is popped, and a new 64-bit data block can be processed.

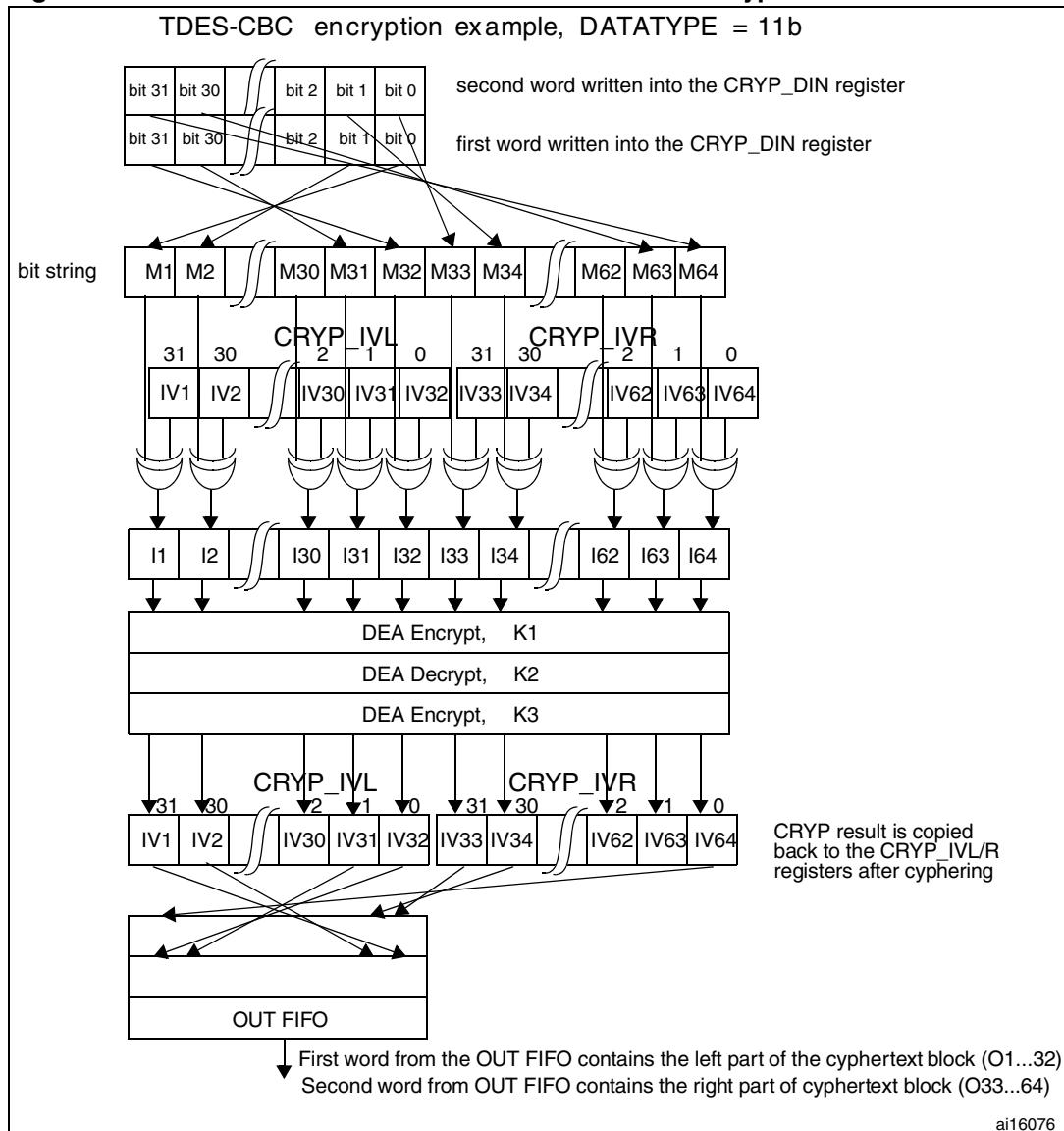
During the AES CBC encryption, the CRYP_IV0...1(L/R) bits are XORed with the 128-bit data block popped off the IN FIFO after swapping (according to the DATATYPE value). When the output of the AES core is available, it is copied back into the CRYP_IV0...1(L/R) vector, and this new content is XORed with the next 128-bit data block popped off the IN FIFO, and so on.

During the AES CBC decryption, the CRYP_IV0...1(L/R) bits are XORed with the 128-bit data block delivered by the AES core before swapping (according to the DATATYPE value) and pushed into the OUT FIFO. When the XORed result is swapped and pushed into the OUT FIFO, the CRYP_IV0...1(L/R) value is replaced by the output of the IN FIFO, then the IN FIFO is popped, and a new 128-bit data block can be processed.

During the AES CTR encryption or decryption, the CRYP_IV0...1(L/R) bits are encrypted by the AES core. Then the result of the encryption is XORed with the 128-bit data block popped off the IN FIFO after swapping (according to the DATATYPE value). When the XORed result is swapped and pushed into the OUT FIFO, the counter part of the CRYP_IV0...1(L/R) value (32 LSB) is incremented.

Any write operation to the CRYP_IV0...1(L/R) registers when bit BUSY = 1b in the CRYP_SR register is disregarded (CRYP_IV0...1(L/R) register content not modified). Thus, you must check that bit BUSY = 0b before modifying initialization vectors.

Figure 202. Initialization vectors use in the TDES-CBC encryption



19.3.5 CRYP busy state

When there is enough data in the input FIFO (at least 2 words for the DES or TDES algorithm mode, 4 words for the AES algorithm mode) and enough free-space in the output FIFO (at least 2 (DES/TDES) or 4 (AES) word locations), and when the bit CRYPEN = 1 in the CRYP_CR register, then the cryptographic processor automatically starts an encryption or decryption process (according to the value of the ALGODIR bit in the CRYP_CR register).

This process takes 48 AHB2 clock cycles for the Triple-DES algorithm, 16 AHB2 clock cycles for the simple DES algorithm, and 14, 16 or 18 AHB2 clock cycles for the AES with key lengths of 128, 192 or 256 bits, respectively. During the whole process, the BUSY bit in the CRYP_SR register is set to 1. At the end of the process, two (DES/TDES) or four (AES) words are written by the CRYP Core into the output FIFO, and the BUSY bit is cleared. In the CBC or CTR mode, the initialization vectors CRYP_IVx(L/R)R (x = 0..3) are updated as well.

A write operation to the key registers (CRYP_Kx(L/R)R, x = 0..3), the initialization registers (CRYP_IVx(L/R)R, x = 0..3), or to bits [9:2] in the CRYP_CR register are ignored when the cryptographic processor is busy (bit BUSY = 1b in the CRYP_SR register), and the registers are not modified. It is thus not possible to modify the configuration of the cryptographic processor while it is processing a block of data. It is however possible to clear the CRYPEN bit while BUSY = 1, in which case the ongoing DES, TDES or AES processing is completed and the two/four word results are written into the output FIFO, and then, only then, the BUSY bit is cleared.

Note: *When a block is being processed in the DES or TDES mode, if the output FIFO becomes full and if the input FIFO contains at least one new block, then the new block is popped off the input FIFO and the BUSY bit remains high until there is enough space to store this new block into the output FIFO.*

19.3.6 Procedure to perform an encryption or a decryption

Initialization

1. Initialize the peripheral (the order of operations is not important except for the key preparation for AES-ECB or AES-CBC decryption. The key size and the key value must be entered before preparing the key and the algorithm must be configured once the key has been prepared):
 - a) Configure the key size (128-, 192- or 256-bit, in the AES only) with the KEYSIZE bits in the CRYP_CR register
 - b) Write the symmetric key into the CRYP_KxL/R registers (2 to 8 registers to be written depending on the algorithm)
 - c) Configure the data type (1-, 8-, 16- or 32-bit), with the DATATYPE bits in the CRYP_CR register
 - d) In case of decryption in AES-ECB or AES-CBC, you must prepare the key: configure the key preparation mode by setting the ALGOMODE bits to '111' in the CRYP_CR register. Then write the CRYPEN bit to 1: the BUSY bit is set. Wait until BUSY returns to 0 (CRYPEN is automatically cleared as well): the key is prepared for decryption
 - e) Configure the algorithm and chaining (the DES/TDES in ECB/CBC, the AES in ECB/CBC/CTR) with the ALGOMODE bits in the CRYP_CR register
 - f) Configure the direction (encryption/decryption), with the ALGODIR bit in the CRYP_CR register
 - g) Write the initialization vectors into the CRYP_IVxL/R register (in CBC or CTR modes only)
2. Flush the IN and OUT FIFOs by writing the FFLUSH bit to 1 in the CRYP_CR register

Processing when the DMA is used to transfer the data from/to the memory

1. Configure the DMA controller to transfer the input data from the memory. The transfer length is the length of the message. As message padding is not managed by the peripheral, the message length must be an entire number of blocks. The data are transferred in burst mode. The burst length is 4 words in the AES and 2 or 4 words in the DES/TDES. The DMA should be configured to set an interrupt on transfer completion of the output data to indicate that the processing is finished.
2. Enable the cryptographic processor by writing the CRYPEN bit to 1. Enable the DMA requests by setting the DIEN and DOEN bits in the CRYP_DMACR register.

3. All the transfers and processing are managed by the DMA and the cryptographic processor. The DMA interrupt indicates that the processing is complete. Both FIFOs are normally empty and BUSY = 0.

Processing when the data are transferred by the CPU during interrupts

1. Enable the interrupts by setting the INIM and OUTIM bits in the CRYP_IMSCR register.
2. Enable the cryptographic processor by setting the CRYPPEN bit in the CRYP_CR register.
3. In the interrupt managing the input data: load the input message into the IN FIFO. You can load 2 or 4 words at a time, or load data until the FIFO is full. When the last word of the message has been entered into the FIFO, disable the interrupt by clearing the INIM bit.
4. In the interrupt managing the output data: read the output message from the OUT FIFO. You can read 1 block (2 or 4 words) at a time or read data until the FIFO is empty. When the last word has been read, INIM=0, BUSY=0 and both FIFOs are empty (IFEM=1 and OFNE=0). You can disable the interrupt by clearing the OUTIM bit and, the peripheral by clearing the CRYPPEN bit.

Processing without using the DMA nor interrupts

1. Enable the cryptographic processor by setting the CRYPPEN bit in the CRYP_CR register.
2. Write the first blocks in the input FIFO (2 to 8 words).
3. Repeat the following sequence until the complete message has been processed:
 - a) Wait for OFNE=1, then read the OUT-FIFO (1 block or until the FIFO is empty)
 - b) Wait for IFNF=1, then write the IN FIFO (1 block or until the FIFO is full)
4. At the end of the processing, BUSY=0 and both FIFOs are empty (IFEM=1 and OFNE=0). You can disable the peripheral by clearing the CRYPPEN bit.

19.3.7 Context swapping

If a context switching is needed because a new task launched by the OS requires this resource, the following tasks have to be performed for full context restoration (example when the DMA is used):

Case of the AES and DES

1. Context saving
 - a) Stop DMA transfers on the IN FIFO by clearing the DIEN bit in the CRYP_DMACR register.
 - b) Wait until both the IN and OUT FIFOs are empty (IFEM=1 and OFNE=0 in the CRYP_SR register) and the BUSY bit is cleared.
 - c) Stop DMA transfers on the OUT FIFO by writing the DOEN bit to 0 in the CRYP_DMACR register and clear the CRYPPEN bit.
 - d) Save the current configuration (bits [9:2] in the CRYP_CR register) and, if not in ECB mode, the initialization vectors. The key value must already be available in the memory. When needed, save the DMA status (pointers for IN and OUT messages, number of remaining bytes, etc.)
2. Configure and execute the other processing.

3. Context restoration
 - a) Configure the processor as in [Section 19.3.6: Procedure to perform an encryption or a decryption on page 497, Initialization](#) with the saved configuration. For the AES-ECB or AES-CBC decryption, the key must be prepared again.
 - b) If needed, reconfigure the DMA controller to transfer the rest of the message.
 - c) Enable the processor by setting the CRYPEN bit and, the DMA requests by setting the DIEN and DOEN bits.

Case of the TDES

Context swapping can be done in the TDES in the same way as in the AES. But as the input FIFO can contain up to 4 unprocessed blocks and as the processing duration per block is higher, it can be faster in certain cases to interrupt the processing without waiting for the IN FIFO to be empty.

1. Context saving
 - a) Stop DMA transfers on the IN FIFO by clearing the DIEN bit in the CRYP_DMACR register.
 - b) Disable the processor by clearing the CRYPEN bit (the processing will stop at the end of the current block).
 - c) Wait until the OUT FIFO is empty (OFNE=0 in the CRYP_SR register) and the BUSY bit is cleared.
 - d) Stop DMA transfers on the OUT FIFO by writing the DOEN bit to 0 in the CRYP_DMACR register.
 - e) Save the current configuration (bits [9:2] in the CRYP_CR register) and, if not in ECB mode, the initialization vectors. The key value must already be available in the memory. When needed, save the DMA status (pointers for IN and OUT messages, number of remaining bytes, etc.). Read back the data loaded in the IN FIFO that have not been processed and save them in the memory until the FIFO is empty.
2. Configure and execute the other processing.
3. Context restoration
 - a) Configure the processor as in [Section 19.3.6: Procedure to perform an encryption or a decryption on page 497, Initialization](#) with the saved configuration. For the AES-ECB or AES-CBC decryption, the key must be prepared again.
 - b) Write the data that were saved during context saving into the IN FIFO.
 - c) If needed, reconfigure the DMA controller to transfer the rest of the message.
 - d) Enable the processor by setting the CRYPEN bit and, the DMA requests by setting the DIEN and DOEN bits.

19.4 CRYP interrupts

There are two individual maskable interrupt sources generated by the CRYP. These two sources are combined into a single interrupt signal, which is the only interrupt signal from the CRYP that drives the NVIC (nested vectored interrupt controller). This combined interrupt, which is an OR function of the individual masked sources, is asserted if any of the individual interrupts listed below is asserted and enabled.

You can enable or disable the interrupt sources individually by changing the mask bits in the CRYP_IMSCR register. Setting the appropriate mask bit to 1 enables the interrupt.

The status of the individual interrupt sources can be read either from the CRYP_RISR register, for raw interrupt status, or from the CRYP_MISR register, for the masked interrupt status.

Output FIFO service interrupt - OUTMIS

The output FIFO service interrupt is asserted when there is one or more (32-bit word) data items in the output FIFO. This interrupt is cleared by reading data from the output FIFO until there is no valid (32-bit) word left (that is, the interrupt follows the state of the OFNE (output FIFO not empty) flag).

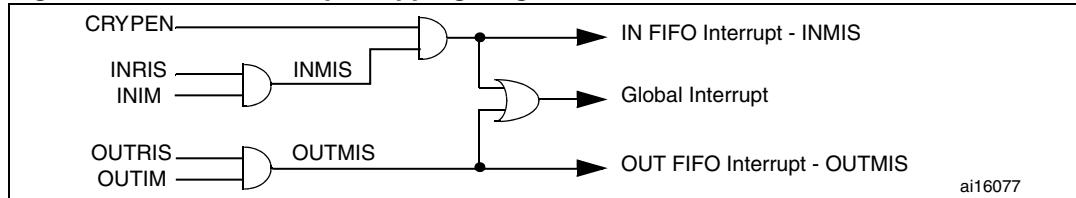
The output FIFO service interrupt OUTMIS is NOT enabled with the CRYP enable bit. Consequently, disabling the CRYP will not force the OUTMIS signal low if the output FIFO is not empty.

Input FIFO service interrupt - INMIS

The input FIFO service interrupt is asserted when there are less than four words in the input FIFO. It is cleared by performing write operations to the input FIFO until it holds four or more words.

The input FIFO service interrupt INMIS is enabled with the CRYP enable bit. Consequently, when CRYP is disabled, the INMIS signal is low even if the input FIFO is empty.

Figure 203. CRYP interrupt mapping diagram



19.5 CRYP DMA interface

The cryptographic processor provides an interface to connect to the DMA controller. The DMA operation is controlled through the CRYP DMA control register, CRYP_DMACR.

The burst and single transfer request signals are not mutually exclusive. They can both be asserted at the same time. For example, when there are 6 words available in the OUT FIFO, the burst transfer request and the single transfer request are asserted. After a burst transfer of 4 words, the single transfer request only is asserted to transfer the last 2 available words. This is useful for situations where the number of words left to be received in the stream is less than a burst.

Each request signal remains asserted until the relevant DMA clear signal is asserted. After the request clear signal is deasserted, a request signal can become active again, depending on the above described conditions. All request signals are deasserted if the CRYP peripheral is disabled or the DMA enable bit is cleared (DIEN bit for the IN FIFO and DOEN bit for the OUT FIFO in the CRYP_DMACR register).

- Note:*
- 1 *The DMA controller must be configured to perform burst of 4 words or less. Otherwise some data could be lost.*
 - 2 *In order to let the DMA controller empty the OUT FIFO before filling up the IN FIFO, the OUTDMA channel should have a higher priority than the INDMA channel.*

19.6 CRYP registers

The cryptographic core is associated with several control and status registers, eight key registers and four initialization vectors registers.

19.6.1 CRYP control register (CRYP_CR)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit 31:16 Reserved, must be kept at reset value

Bit 15 **CRYPEN**: Cryptographic processor enable

- 0: CRYP processor is disabled
- 1: CRYP processor is enabled

Note: The CRYPEN bit is automatically cleared by hardware when the key preparation process ends (ALGOMODE=111b).

Bit 14 **FFLUSH**: FIFO flush

When CRYPEN = 0, writing this bit to 1 flushes the IN and OUT FIFOs (that is read and write pointers of the FIFOs are reset. Writing this bit to 0 has no effect.

When CRYPEN = 1, writing this bit to 0 or 1 has no effect.

Reading this bit always returns 0.

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

Bits 9:8 **KEYSIZE[1:0]**: Key size selection (AES mode only)

This bitfield defines the bit-length of the key used for the AES cryptographic core. This bitfield is ‘don’t care’ in the DES or TDES modes.

- 00: 128 bit key length
- 01: 192 bit key length
- 10: 256 bit key length
- 11: Reserved, do not use this value

Bits 7:6 **DATATYPE[1:0]**: Data type selection

This bitfield defines the format of data entered in the CRYP_DIN register (refer to [Section 19.3.3: Data type](#)).

00: 32-bit data. No swapping of each word. First word pushed into the IN FIFO (or popped off the OUT FIFO) forms bits 1...32 of the data block, the second word forms bits 33...64.

01: 16-bit data, or half-word. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 2 half-words, which are swapped with each other.

10: 8-bit data, or bytes. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 4 bytes, which are swapped with each other.

11: bit data, or bit-string. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 32 bits (1st bit of the string at position 0), which are swapped with each other.

Bits 5:3 ALGOMODE[2:0]: Algorithm mode

- 000: TDES-ECB (triple-DES Electronic codebook): no feedback between blocks of data. Initialization vectors (CRYP_IV0(L/R)) are not used, three key vectors (K1, K2, and K3) are used (K0 is not used).
- 001: TDES-CBC (triple-DES Cipher block chaining): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R) must be initialized, three key vectors (K1, K2, and K3) are used (K0 is not used).
- 010: DES-ECB (simple DES Electronic codebook): no feedback between blocks of data. Initialization vectors (CRYP_IV0L/R) are not used, only one key vector (K1) is used (K0, K2, K3 are not used).
- 011: DES-CBC (simple DES Cipher block chaining): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R) must be initialized. Only one key vector (K1) is used (K0, K2, K3 are not used).
- 100: AES-ECB (AES Electronic codebook): no feedback between blocks of data. Initialization vectors (CRYP_IV0L/R...1L/R) are not used. All four key vectors (K0...K3) are used.
- 101: AES-CBC (AES Cipher block chaining): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R...1L/R) must be initialized. All four key vectors (K0...K3) are used.
- 110: AES-CTR (AES Counter mode): output block is XORed with the subsequent input block before its entry into the algorithm. Initialization vectors (CRYP_IV0L/R...1L/R) must be initialized. All four key vectors (K0...K3) are used. CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that will be XORed with the plaintext or cipher in input. Thus, ALGODIR is don't care when ALGOMODE = 110b, and the key must NOT be unrolled (prepared) for decryption.
- 111: AES key preparation for decryption mode. Writing this value when CRYPPEN = 1 immediately starts an AES round for key preparation. The secret key must have previously been loaded into the K0...K3 registers. The BUSY bit in the CRYP_SR register is set during the key preparation. After key processing, the resulting key is copied back into the K0...K3 registers, and the BUSY bit is cleared.

Bit 2 ALGODIR: Algorithm direction

- 0: Encrypt
1: Decrypt

Bit 1:0 Reserved, must be kept at reset value

Note: Writing to the KEYSIZE, DATATYPE, ALGOMODE and ALGODIR bits while BUSY=1 has no effect. These bits can only be configured when BUSY=0.

The FFLUSH bit has to be set only when BUSY=0. If not, the FIFO is flushed, but the block being processed may be pushed into the output FIFO just after the flush operation, resulting in a nonempty FIFO condition.

19.6.2 CRYP status register (CRYP_SR)

Address offset: 0x04

Reset value: 0x0000 0003

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

Bit 31:5 Reserved, must be kept at reset value

Bit 4 **BUSY**: Busy bit

0: The CRYP Core is not processing any data. The reason is either that:

- the CRYP core is disabled (CRYPEN=0 in the CRYP_CR register) and the last processing has completed, or
- The CRYP core is waiting for enough data in the input FIFO or enough free space in the output FIFO (that is in each case at least 2 words in the DES, 4 words in the AES).

1: The CRYP core is currently processing a block of data or a key preparation (for AES decryption).

Bit 3 **OFFU**: Output FIFO full

0: Output FIFO is not full

1: Output FIFO is full

Bits 2 **OFNE**: Output FIFO not empty

0: Output FIFO is empty

1: Output FIFO is not empty

Bit 1 **IFNF**: Input FIFO not full

0: Input FIFO is full

1: Input FIFO is not full

Bits 0 **IFEM**: Input FIFO empty

0: Input FIFO is not empty

1: Input FIFO is empty

19.6.3 CRYP data input register (CRYP_DIN)

Address offset: 0x08

Reset value: 0x0000 0000

The CRYP_DIN is the data input register. It is 32-bit wide. It is used to enter up to four 64-bit (TDES) or two 128-bit (AES) plaintext (when encrypting) or ciphertext (when decrypting) blocks into the input FIFO, one 32-bit word at a time.

The first word written into the FIFO is the MSB of the input block. The LSB of the input block is written at the end. Disregarding the data swapping, this gives:

- In the DES/TDES modes: a block is a sequence of bits numbered from bit 1 (leftmost bit) to bit 64 (rightmost bit). Bit 1 corresponds to the MSB (bit 31) of the first word entered into the FIFO, bit 64 corresponds to the LSB (bit 0) of the second word entered into the FIFO.
- In the AES mode: a block is a sequence of bits numbered from 0 (leftmost bit) to 127 (rightmost bit). Bit 0 corresponds to the MSB (bit 31) of the first word written into the FIFO, bit 127 corresponds to the LSB (bit 0) of the 4th word written into the FIFO.

To fit different data sizes, the data written in the CRYP_DIN register can be swapped before being processed by configuring the DATATYPE bits in the CRYP_CR register. Refer to [Section 19.3.3: Data type on page 492](#) for more details.

When CRYP_DIN is written to, the data are pushed into the input FIFO. When at least two 32-bit words in the DES/TDES mode (or four 32-bit words in the AES mode) have been pushed into the input FIFO, and when at least 2 words are free in the output FIFO, the CRYP engine starts an encrypting or decrypting process. This process takes two 32-bit words in the DES/TDES mode (or four 32-bit words in the AES mode) from the input FIFO and delivers two 32-bit words (or 4, respectively) to the output FIFO per process round.

When CRYP_DIN is read:

- If CRYPEN = 0, the FIFO is popped, and then the data present in the Input FIFO are returned, from the oldest one (first reading) to the newest one (last reading). The IFEM flag must be checked before each read operation to make sure that the FIFO is not empty.
- if CRYPEN = 1, an undefined value is returned.

After the CRYP_DIN register has been read once or several times, the FIFO must be flushed by setting the FFLUSH bit prior to processing new data.

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

Bit 31:0 **DATAIN**: Data input

Read = returns Input FIFO content if CRYPEN = 0, else returns an undefined value.
Write = Input FIFO is written.

19.6.4 CRYP data output register (CRYP_DOUT)

Address offset: 0x0C

Reset value: 0x0000 0000

The CRYP_DOUT is the data output register. It is read-only and 32-bit wide. It is used to retrieve up to four 64-bit (TDES mode) or two 128-bit (AES mode) ciphertext (when encrypting) or plaintext (when decrypting) blocks from the output FIFO, one 32-bit word at a time.

Like for the input data, the MSB of the output block is the first word read from the output FIFO. The LSB of the output block is read at the end. Disregarding data swapping, this gives:

- In the DES/TDES modes: Bit 1 (leftmost bit) corresponds to the MSB (bit 31) of the first word read from the FIFO, bit 64 (rightmost bit) corresponds to the LSB (bit 0) of the second word read from the FIFO.
- In the AES mode: Bit 0 (leftmost bit) corresponds to the MSB (bit 31) of the first word read from the FIFO, bit 127 (rightmost bit) corresponds to the LSB (bit 0) of the 4th word read from the FIFO.

To fit different data sizes, the data can be swapped after processing by configuring the DATATYPE bits in the CRYP_CR register. Refer to [Section 19.3.3: Data type on page 492](#) for more details.

When CRYP_DOUT is read, the last data entered into the output FIFO (pointed to by the read pointer) is returned.

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

Bit 31:0 **DATAOUT**: Data output

Read = returns output FIFO content.

Write = No effect.

19.6.5 CRYP DMA control register (CRYP_DMCR)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit 31:2 Reserved, must be kept at reset value

Bit 1 **DOEN**: DMA output enable

- 0: DMA for outgoing data transfer is disabled
- 1: DMA for outgoing data transfer is enabled

Bit 0 **DIEN**: DMA input enable

- 0: DMA for incoming data transfer is disabled
- 1: DMA for incoming data transfer is enabled

19.6.6 CRYP interrupt mask set/clear register (CRYP_IMSCR)

Address offset: 0x14

Reset value: 0x0000 0000

The CRYP_IMSCR register is the interrupt mask set or clear register. It is a read/write register. On a read operation, this register gives the current value of the mask on the relevant interrupt. Writing 1 to the particular bit sets the mask, enabling the interrupt to be read. Writing 0 to this bit clears the corresponding mask. All the bits are cleared to 0 when the peripheral is reset.

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

Bit 31:2 Reserved, must be kept at reset value

Bit 1 **OUTIM**: Output FIFO service interrupt mask

- 0: Output FIFO service interrupt is masked
- 1: Output FIFO service interrupt is not masked

Bit 0 **INIM**: Input FIFO service interrupt mask

- 0: Input FIFO service interrupt is masked
- 1: Input FIFO service interrupt is not masked

19.6.7 CRYP raw interrupt status register (CRYP_RISR)

Address offset: 0x18

Reset value: 0x0000 0001

The CRYP_RISR register is the raw interrupt status register. It is a read-only register. On a read, this register gives the current raw status of the corresponding interrupt prior to masking. A write has no effect.

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

Bit 31:2 Reserved, must be kept at reset value

Bit 1 **OUTRIS:** Output FIFO service raw interrupt status

Gives the raw interrupt state prior to masking of the output FIFO service interrupt.

- 0: Raw interrupt not pending
- 1: Raw interrupt pending

Bit 0 **INRIS:** Input FIFO service raw interrupt status

Gives the raw interrupt state prior to masking of the Input FIFO service interrupt.

- 0: Raw interrupt not pending
- 1: Raw interrupt pending

19.6.8 CRYP masked interrupt status register (CRYP_MISR)

Address offset: 0x1C

Reset value: 0x0000 0000

The CRYP_MISR register is the masked interrupt status register. It is a read-only register. On a read, this register gives the current masked status of the corresponding interrupt prior to masking. A write has no effect.

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

Bit 31:2 Reserved, must be kept at reset value

Bit 1 **OUTMIS:** Output FIFO service masked interrupt status

Gives the interrupt state after masking of the output FIFO service interrupt.

- 0: Interrupt not pending
- 1: Interrupt pending

Bit 0 **INMIS**: Input FIFO service masked interrupt status

Gives the interrupt state after masking of the input FIFO service interrupt.

0: Interrupt not pending

1: Interrupt pending when CRYPEN = 1

19.6.9 CRYP key registers (CRYP_K0...3(L/R)R)

Address offset: 0x20 to 0x3C

Reset value: 0x0000 0000

These registers contain the cryptographic keys.

In the TDES mode, keys are 64-bit binary values (number from left to right, that is the leftmost bit is bit 1), named K1, K2 and K3 (K0 is not used), each key consists of 56 information bits and 8 parity bits. The parity bits are reserved for error detection purposes and are not used by the current block. Thus, bits 8, 16, 24, 32, 40, 48, 56 and 64 of each 64-bit key value $K_x[1:64]$ are not used.

In the AES mode, the key is considered as a single 128-, 192- or 256-bit long bit sequence, $k_0k_1k_2\dots k_{127/191/255}$ (k_0 being the leftmost bit). The AES key is entered into the registers as follows:

- for AES-128: $k_0..k_{127}$ corresponds to $b_{127}..b_0$ ($b_{255}..b_{128}$ are not used),
- for AES-192: $k_0..k_{191}$ corresponds to $b_{191}..b_0$ ($b_{255}..b_{192}$ are not used),
- for AES-256: $k_0..k_{255}$ corresponds to $b_{255}..b_0$.

In any case b_0 is the rightmost bit.

CRYP_K0LR (address offset: 0x20)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
b255	b254	b253	b252	b251	b250	b249	b248	b247	b246	b245	b244	b243	b242	b241	b240
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
b239	b238	b237	b236	b235	b234	b233	b232	b231	b230	b229	b228	b227	b226	b225	b224
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K0RR (address offset: 0x24)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
b223	b222	b221	b220	b219	b218	b217	b216	b215	b214	b213	b212	b211	b210	b209	b208
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
b207	b206	b205	b204	b203	b202	b201	b200	b199	b198	b197	b196	b195	b194	b193	b192
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K1LR (address offset: 0x28)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k1.1 b191	k1.2 b190	k1.3 b189	k1.4 b188	k1.5 b187	k1.6 b186	k1.7 b185	k1.8 b184	k1.9 b183	k1.10 b182	k1.11 b181	k1.12 b180	k1.13 b179	k1.14 b178	k1.15 b177	k1.16 b176
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
k1.17 b175	k1.18 b174	k1.19 b173	k1.20 b172	k1.21 b171	k1.22 b170	k1.23 b169	k1.24 b168	k1.25 b167	k1.26 b166	k1.27 b165	k1.28 b164	k1.29 b163	k1.30 b162	k1.31 b161	k1.32 b160
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K1RR (address offset: 0x2C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k1.33 b159	k1.34 b158	k1.35 b157	k1.36 b156	k1.37 b155	k1.38 b154	k1.39 b153	k1.40 b152	k1.41 b151	k1.42 b150	k1.43 b149	k1.44 b148	k1.45 b147	k1.46 b146	k1.47 b145	k1.48 b144
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
k1.49 b143	k1.50 b142	k1.51 b141	k1.52 b140	k1.53 b139	k1.54 b138	k1.55 b137	k1.56 b136	k1.57 b135	k1.58 b134	k1.59 b133	k1.60 b132	k1.61 b131	k1.62 b130	k1.63 b129	k1.64 b128
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K2LR (address offset: 0x30)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k2.1 b127	k2.2 b126	k2.3 b125	k2.4 b124	k2.5 b123	k2.6 b122	k2.7 b121	k2.8 b120	k2.9 b119	k2.10 b118	k2.11 b117	k2.12 b116	k2.13 b115	k2.14 b114	k2.15 b113	k2.16 b112
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
k2.17 b111	k2.18 b110	k2.19 b109	k2.20 b108	k2.21 b107	k2.22 b106	k2.23 b105	k2.24 b104	k2.25 b103	k2.26 b102	k2.27 b101	k2.28 b100	k2.29 b99	k2.30 b98	k2.31 b97	k2.32 b96
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K2RR (address offset: 0x34)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k2.33 b95	k2.34 b94	k2.35 b93	k2.36 b92	k2.37 b91	k2.38 b90	k2.39 b89	k2.40 b88	k2.41 b87	k2.42 b86	k2.43 b85	k2.44 b84	k2.45 b83	k2.46 b82	k2.47 b81	k2.48 b80
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
k2.49 b79	k2.50 b78	k2.51 b77	k2.52 b76	k2.53 b75	k2.54 b74	k2.55 b73	k2.56 b72	k2.57 b71	k2.58 b70	k2.59 b69	k2.60 b68	k2.61 b67	k2.62 b66	k2.63 b65	k2.64 b64
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K3LR (address offset: 0x38)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k3.1 b63	k3.2 b62	k3.3 b61	k3.4 b60	k3.5 b59	k3.6 b58	k3.7 b57	k3.8 b56	k3.9 b55	k3.10 b54	k3.11 b53	k3.12 b52	k3.13 b51	k3.14 b50	k3.15 b49	k3.16 b48
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
k3.17 b47	k3.18 b46	k3.19 b45	k3.20 b44	k3.21 b43	k3.22 b42	k3.23 b41	k3.24 b40	k3.25 b39	k3.26 b38	k3.27 b37	k3.28 b36	k3.29 b35	k3.30 b34	k3.31 b33	k3.32 b32
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K3RR (address offset: 0x3C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k3.33 b31	k3.34 b30	k3.35 b29	k3.36 b28	k3.37 b27	k3.38 b26	k3.39 b25	k3.40 b24	k3.41 b23	k3.42 b22	k3.43 b21	k3.44 b20	k3.45 b19	k3.46 b18	k3.47 b17	k3.48 b16
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
k3.49 b15	k3.50 b14	k3.51 b13	k3.52 b12	k3.53 b11	k3.54 b10	k3.55 b9	k3.56 b8	k3.57 b7	k3.58 b6	k3.59 b5	k3.60 b4	k3.61 b3	k3.62 b2	k3.63 b1	k3.64 b0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Note: Write accesses to these registers are disregarded when the cryptographic processor is busy (bit BUSY = 1 in the CRYP_SR register).

19.6.10 CRYP initialization vector registers (CRYP_IV0...1(L/R)R)

Address offset: 0x40 to 0x4C

Reset value: 0x0000 0000

The CRYP_IV0...1(L/R)R are the left-word and right-word registers for the initialization vector (64 bits for DES/TDES and 128 bits for AES) and are used in the CBC (Cipher block chaining) and Counter (CTR) modes. After each computation round of the TDES or AES Core, the CRYP_IV0...1(L/R)R registers are updated as described in [Section : DES and TDES Cipher block chaining \(DES/TDES-CBC\) mode on page 483](#), [Section : AES Cipher block chaining \(AES-CBC\) mode on page 487](#) and [Section : AES counter mode \(AES-CTR\) mode on page 489](#).

IV0 is the leftmost bit whereas IV63 (DES, TDES) or IV127 (AES) are the rightmost bits of the initialization vector. IV1(L/R)R is used only in the AES.

CRYP_IV0LR (address offset: 0x40)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV0	IV1	IV2	IV3	IV4	IV5	IV6	IV7	IV8	IV9	IV10	IV11	IV12	IV13	IV14	IV15
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV16	IV17	IV18	IV19	IV20	IV21	IV22	IV23	IV24	IV25	IV26	IV27	IV28	IV29	IV30	IV31
rw															

CRYP_IV0RR (address offset: 0x44)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV32	IV33	IV34	IV35	IV36	IV37	IV38	IV39	IV40	IV41	IV42	IV43	IV44	IV45	IV46	IV47
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV48	IV49	IV50	IV51	IV52	IV53	IV54	IV55	IV56	IV57	IV58	IV59	IV60	IV61	IV62	IV63
rw															

CRYP_IV1LR (address offset: 0x48)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV64	IV65	IV66	IV67	IV68	IV69	IV70	IV71	IV72	IV73	IV74	IV75	IV76	IV77	IV78	IV79
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV80	IV81	IV82	IV83	IV84	IV85	IV86	IV87	IV88	IV89	IV90	IV91	IV92	IV93	IV94	IV95
rw															

CRYP_IV1RR (address offset: 0x4C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV96	IV97	IV98	IV99	IV100	IV101	IV102	IV103	IV104	IV105	IV106	IV107	IV108	IV109	IV110	IV111
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV112	IV113	IV114	IV115	IV116	IV117	IV118	IV119	IV120	IV121	IV122	IV123	IV124	IV125	IV126	IV127
rw															

Note: In DES/3DES modes, only CRYP_IV0(L/R) is used.

Note: Write access to these registers are disregarded when the cryptographic processor is busy (bit BUSY = 1 in the CRYP_SR register).

19.6.11 CRYP register map

Table 73. CRYP register map and reset values

Offset	Register name reset value	Register size															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x00	CRYP_CR 0x0000000	Reserved															
0x04	CRYP_SR 0x0000003	Reserved															
0x08	CRYP_DR 0x0000000	DATAIN															
0x0C	CRYP_DOUT 0x0000000	DATAOUT															
0x10	CRYP_DMACR 0x0000000	Reserved															
0x14	CRYP_IMSCR 0x0000000	Reserved															
0x18	CRYP_RISR 0x0000001	Reserved															
0x1C	CRYP_MISR 0x0000000	Reserved															
0x20	CRYP_K0LR 0x0000000	CRYP_K0LR															
0x24	CRYP_K0RR 0x0000000	CRYP_K0RR															
...																	
0x38	CRYP_K3LR 0x0000000	CRYP_K3LR															

Table 73. CRYP register map and reset values (continued)

Refer to [Table 1 on page 50](#) for the register boundary addresses.

20 Random number generator (RNG)

20.1 RNG introduction

The RNG processor is a random number generator, based on a continuous analog noise, that provides a random 32-bit value to the host when read. The RNG is expected to provide a success ratio of more than 85% to FIPS 140-2 tests for a sequence of 20 000 bits, measured on corner conditions by device characterization.

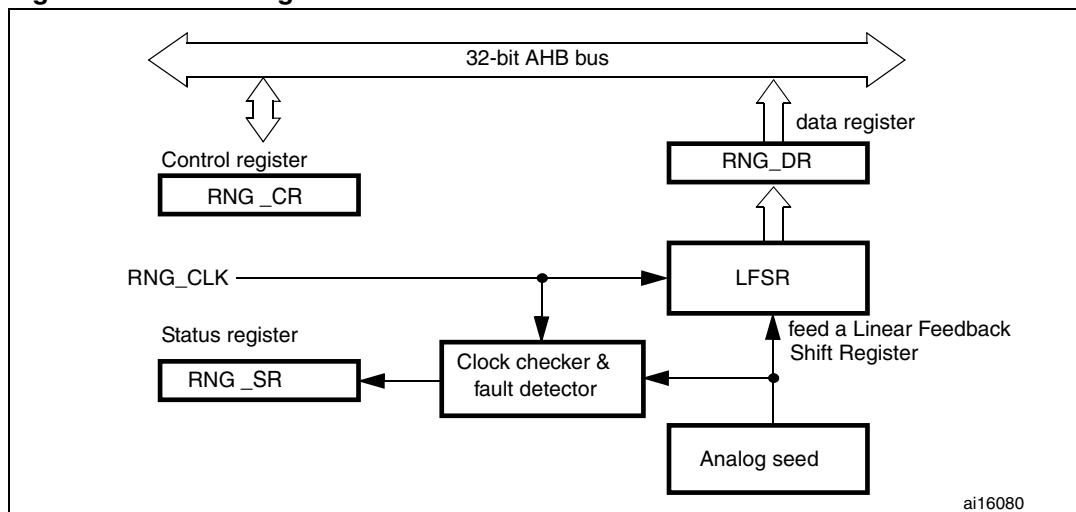
20.2 RNG main features

- It delivers 32-bit random numbers, produced by an analog generator
- 40 periods of the PLL48CLK clock signal between two consecutive random numbers
- Monitoring of the RNG entropy to flag abnormal behavior (generation of stable values, or of a stable sequence of values)
- It can be disabled to reduce power-consumption

20.3 RNG functional description

Figure 204 shows the RNG block diagram.

Figure 204. Block diagram



The random number generator implements an analog circuit. This circuit generates seeds that feed a linear feedback shift register (RNG_LFSR) in order to produce 32-bit random numbers.

The analog circuit is made of several ring oscillators whose outputs are XORed to generate the seeds. The RNG_LFSR is clocked by a dedicated clock (PLL48CLK) at a constant frequency, so that the quality of the random number is independent of the HCLK frequency. The contents of the RNG_LFSR are transferred into the data register (RNG_DR) when a significant number of seeds have been introduced into the RNG_LFSR.

In parallel, the analog seed and the dedicated PLL48CLK clock are monitored. Status bits (in the RNG_SR register) indicate when an abnormal sequence occurs on the seed or when the frequency of the PLL48CLK clock is too low. An interrupt can be generated when an error is detected.

20.3.1 Operation

To run the RNG, follow the steps below:

1. Enable the interrupt if needed (to do so, set the IE bit in the RNG_CR register). An interrupt is generated when a random number is ready or when an error occurs.
2. Enable the random number generation by setting the RNGEN bit in the RNG_CR register. This activates the analog part, the RNG_LFSR and the error detector.
3. At each interrupt, check that no error occurred (the SEIS and CEIS bits should be '0' in the RNG_SR register) and that a random number is ready (the DRDY bit is '1' in the RNG_SR register). The contents of the RNG_DR register can then be read.

As required by the FIPS PUB (Federal Information Processing Standard Publication) 140-2, the first random number generated after setting the RNGEN bit should not be used, but saved for comparison with the next generated random number. Each subsequent generated random number has to be compared with the previously generated number. The test fails if any two compared numbers are equal (continuous random number generator test).

20.3.2 Error management

If the CEIS bit is read as '1' (clock error)

In the case of a clock, the RNG is no more able to generate random numbers because the PLL48CLK clock is not correct. Check that the clock controller is correctly configured to provide the RNG clock and clear the CEIS bit. The RNG can work when the CECS bit is '0'. The clock error has no impact on the previously generated random numbers, and the RNG_DR register contents can be used.

If the SEIS bit is read as '1' (seed error)

In the case of a seed error, the generation of random numbers is interrupted for as long as the SECS bit is '1'. If a number is available in the RNG_DR register, it must not be used because it may not have enough entropy.

What you should do is clear the SEIS bit, then clear and set the RNGEN bit to reinitialize and restart the RNG.

20.4 RNG registers

The RNG is associated with a control register, a data register and a status register.

20.4.1 RNG control register (RNG_CR)

Address offset: 0x00

Reset value: 0x0000 0000

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

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

Bit 3 **IE:** Interrupt enable

- 0: RNG Interrupt is disabled
- 1: RNG Interrupt is enabled. An interrupt is pending as soon as DRDY=1 or SEIS=1 or CEIS=1 in the RNG_SR register.

Bit 2 **RNGEN:** Random number generator enable

- 0: Random number generator is disabled
- 1: random Number Generator is enabled.

Bits 1:0 Reserved, must be kept at reset value

20.4.2 RNG status register (RNG_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
												SEIS	CEIS	Reserved	
												rc_w0	rc_w0		
												SECS	CECS	DRDY	
												r	r	r	

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

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS, it is cleared by writing it to 0.

- 0: No faulty sequence detected
- 1: One of the following faulty sequences has been detected:
 - More than 64 consecutive bits at the same value (0 or 1)
 - More than 32 consecutive alternances of 0 and 1 (0101010101...01)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS, it is cleared by writing it to 0.

- 0: The PLL48CLK clock was correctly detected
- 1: The PLL48CLK was not correctly detected ($f_{PLL48CLK} < f_{HCLK}/16$)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: One of the following faulty sequences has been detected:

- More than 64 consecutive bits at the same value (0 or 1)
- More than 32 consecutive alternances of 0 and 1 (0101010101...01)

Bit 1 **CECS:** Clock error current status

0: The PLL48CLK clock has been correctly detected. If the CEIS bit is set, this means that a clock error was detected and the situation has been recovered

1: The PLL48CLK was not correctly detected ($f_{PLL48CLK} < f_{HCLK}/16$).

Bit 0 **DRDY:** Data ready

0: The RNG_DR register is not yet valid, no random data is available

1: The RNG_DR register contains valid random data

Note: An interrupt is pending if IE = 1 in the RNG_CR register.

Once the RNG_DR register has been read, this bit returns to 0 until a new valid value is computed.

20.4.3 RNG data register (RNG_DR)

Address offset: 0x08

Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. After being read, this register delivers a new random value after a maximum time of 40 periods of the PLL48CLK clock. The software must check that the DRDY bit is set before reading the RNDATA value.

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

Bits 31:0 **RNDATA:** Random data

32-bit random data.

20.4.4 RNG register map

Table 74 gives the RNG register map and reset values.

Table 74. RNG register map and reset map

Offset	Register name reset value	Register size																															
		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	RNG_CR 0x00000000																																
0x04	RNG_SR 0x00000000																																
0x08	RNG_DR 0x00000000																																

21 Hash processor (HASH)

21.1 HASH introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1), the MD5 (message-digest algorithm 5) hash algorithm and the HMAC (keyed-hash message authentication code) algorithm suitable for a variety of applications. It computes a message digest (160 bits for the SHA-1 algorithm, 128 bits for the MD5 algorithm) for messages of up to $(2^{64} - 1)$ bits, while HMAC algorithms provide a way of authenticating messages by means of hash functions. HMAC algorithms consist in calling the SHA-1 or MD5 hash function twice.

21.2 HASH main features

- Suitable for data authentication applications, compliant with:
 - FIPS PUB 180-2 (Federal Information Processing Standards Publication 180-2)
 - Secure Hash Standard specifications (SHA-1)
 - IETF RFC 1321 (Internet Engineering Task Force Request For Comments number 1321) specifications (MD5)
- AHB slave peripheral
- 32-bit data words for input data, supporting word, half-word, byte and bit bit-string representations, with little-endian data representation only
- Automatic swapping to comply with the big-endian SHA1 computation standard with little-endian input bit-string representation
- Automatic padding to complete the input bit string to fit modulo 512 (16×32 bits) message digest computing
- Fast computation of SHA-1 and MD5
- 5×32 -bit words (H_0, H_1, H_2, H_3 and H_4) for output message digest, reload able to continue interrupted message digest computation
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
- Automatic data flow control with support for direct memory access (DMA)

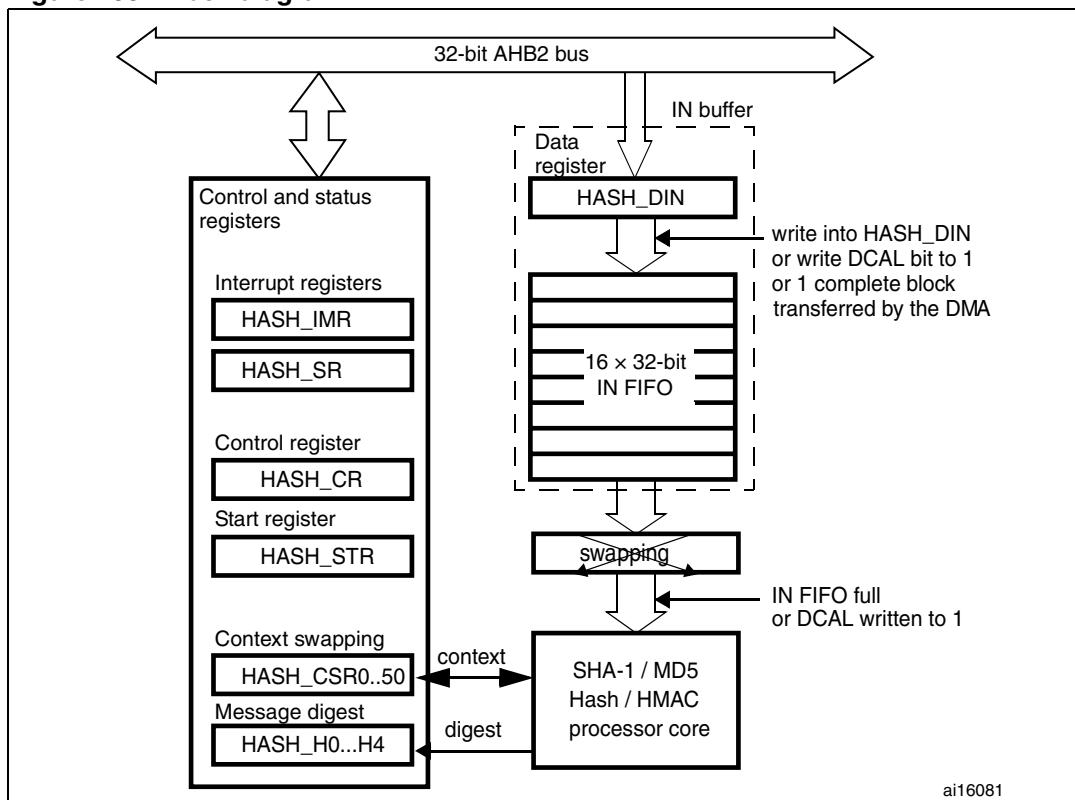
Note: *Padding, as defined in the SHA-1 algorithm, consists in adding a bit at b_{x1} followed by N bits at b_{x0} to get a total length congruent to 448 modulo 512. After this, the message is completed with a 64-bit integer which is the binary representation of the original message length.*

For this hash processor, the quanta for entering the message is a 32-bit word, so an additional information must be provided at the end of the message entry, which is the number of valid bits in the last 32-bit word entered.

21.3 HASH functional description

Figure 205 shows the block diagram of the hash processor.

Figure 205. Block diagram



The FIPS PUB 180-2 standard and the IETF RFC 1321 publication specify the SHA-1 and MD5 secure hash algorithms, respectively, for computing a condensed representation of a message or data file. When a message of any length below 2^{64} bits is provided on input, the SHA-1 and MD5 produce a 160-bit and 128-bit output string, respectively, called a message digest. The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message. Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-1 and MD5 are qualified as “secure” because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify. For more detail on the SHA-1 algorithm, please refer to the FIPS PUB 180-2 (Federal Information Processing Standards Publication 180-2), 2002 august 1.

The current implementation of this standard works with little-endian input data convention. For example, the C string “abc” must be represented in memory as the 24-bit hexadecimal value 0x434241.

A message or data file to be processed by the hash processor should be considered a bit string. The length of the message is the number of bits in the message (the empty message

has length 0). You can consider that 32 bits of this bit string forms a 32-bit word. Note that the FIPS PUB 180-1 standard uses the convention that bit strings grow from left to right, and bits can be grouped as bytes (8 bits) or words (32 bits) (but some implementations also use half-words (16 bits), and implicitly, uses the big-endian byte (half-word) ordering. This convention is mainly important for padding (see [Section 21.3.4: Message padding on page 522](#)).

21.3.1 Duration of the processing

The computation of an intermediate block of a message takes:

- 66 HCLK clock cycles in SHA-1
- 50 HCLK clock cycles in MD5

to which you must add the time needed to load the 16 words of the block into the processor (at least 16 clock cycles for a 512-bit block).

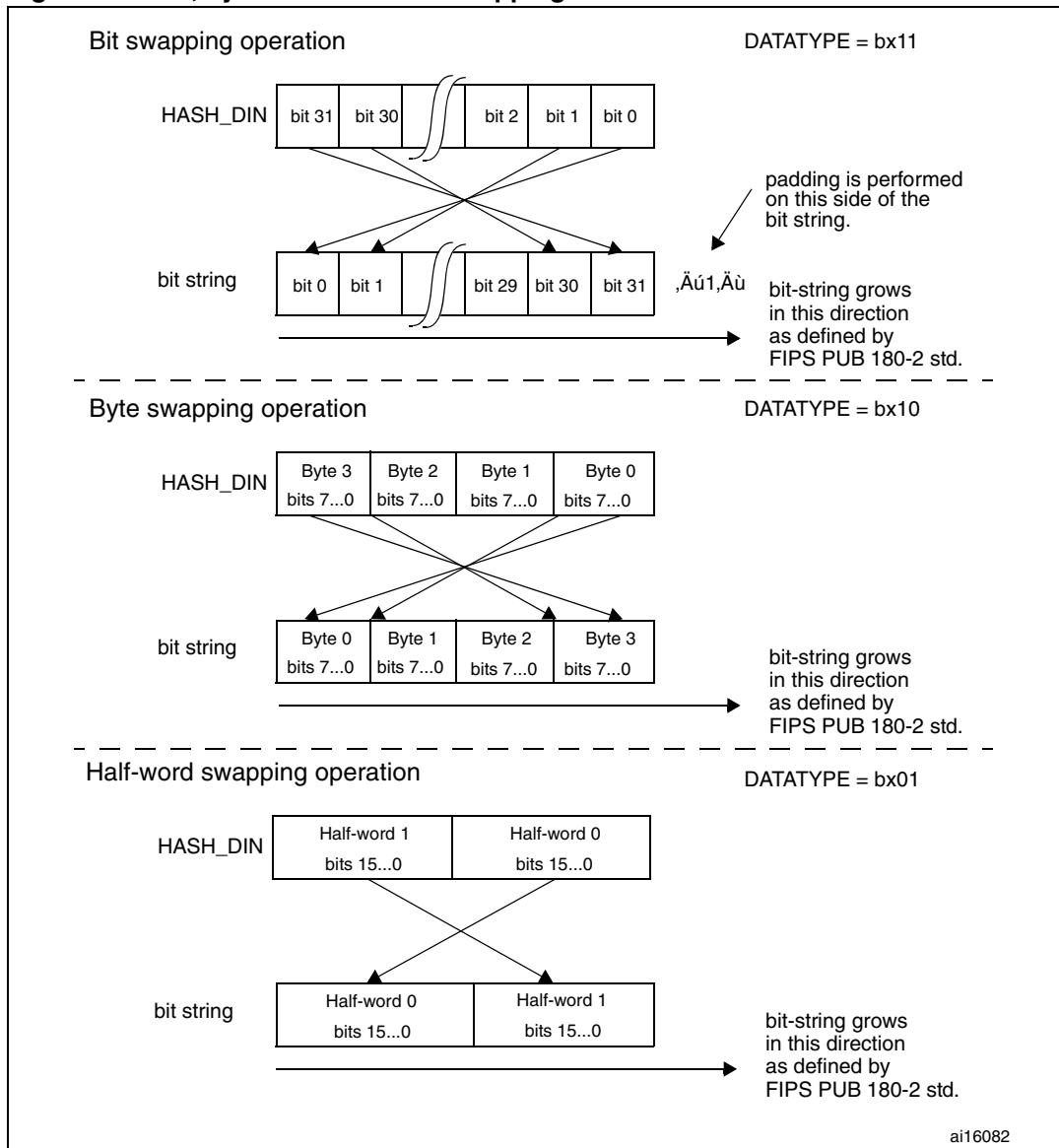
The time needed to process the last block of a message (or of a key in HMAC) can be longer because it includes the padding. This time depends on the length of the last block and the size of the key (in HMAC mode). Compared to the processing of an intermediate block, it can be increased by a factor of:

- 1 to 2.5 for a hash message
- around 2.5 for an HMAC input-key
- 1 to 2.5 for an HMAC message
- around 2.5 for an HMAC output key in case of a short key
- 3.5 to 5 for an HMAC output key in case of a long key

21.3.2 Data type

Data are entered into the hash processor 32 bits (word) at a time, by writing them into the HASH_DIN register. But the original bit-string can be organized in bytes, half-words or words, or even be represented as bits. As the system memory organization is little-endian and SHA1 computation is big-endian, depending on the way the original bit string is grouped, a bit, byte, or half-word swapping operation is performed automatically by the hash processor.

The kind of data to be processed is configured with the DATATYPE bitfield in the HASH control register (HASH_CR).

Figure 206. Bit, byte and half-word swapping

The least significant bit of the message has to be at position 0 (right) in the first word entered into the hash processor, the 32nd bit of the bit string has to be at position 0 in the second word entered into the hash processor and so on.

21.3.3 Message digest computing

The HASH sequentially processes blocks of 512 bits when computing the message digest. Thus, each time 16×32 -bit words (= 512 bits) have been written by the DMA or the CPU, into the hash processor, the HASH automatically starts computing the message digest. This operation is known as a partial digest computation.

The message to be processed is entered into the peripheral by 32-bit words written into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the input FIFO (IN FIFO) each time the register is written with new data. HASH_DIN and the input FIFO form a FIFO of a 17-word length (named the IN buffer).

The processing of a block can start only once the last value of the block has entered the IN FIFO. The peripheral must get the information as to whether the HASH_DIN register contains the last bits of the message or not. Two cases may occur:

- When the DMA is not used:
 - In case of a partial digest computation, this is done by writing an additional word into the HASH_DIN register (actually the first word of the next block). Then the software must wait until the processor is ready again (when DINIS=1) before writing new data into HASH_DIN.
 - In case of a final digest computation (last block entered), this is done by writing the DCAL bit to 1.
- When the DMA is used:

The contents of the HASH_DIN register are interpreted automatically with the information sent by the DMA controller.

This process —data entering + partial digest computation— continues until the last bits of the original message are written to the HASH_DIN register. As the length (number of bits) of a message can be any integer value, the last word written into the HASH processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written into the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Once this is done, writing into HASH_STR with bit DCAL = 1 starts the processing of the last entered block of message by the hash processor. This processing consists in:

- Automatically performing the message padding operation: the purpose of this operation is to make the total length of a padded message a multiple of 512. The HASH sequentially processes blocks of 512 bits when computing the message digest
- Computing the final message digest

When the DMA is enabled, it provides the information to the hash processor when it is transferring the last data word. Then the padding and digest computation are performed automatically as if DCAL had been written to 1.

21.3.4 Message padding

Message padding consists in appending a “1” followed by m “0”s followed by a 64-bit integer to the end of the original message to produce a padded message block of length 512. The “1” is added to the last word written into the HASH_DIN register at the bit position defined by the NBLW bitfield, and the remaining upper bits are cleared (“0”s).

Example: let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24:

```
byte 0    byte 1    byte 2    byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a “1” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0”s are appended, making now 448. This gives (in hexadecimal, big-endian format):

```

61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000

```

The L value, in two-word representation (that is 00000000 00000018) is appended. Hence the final padded message in hexadecimal:

```

61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000028

```

If the HASH is programmed to use the little-endian byte input format, the above message has to be entered by doing the following steps:

1. 0xUU636261 is written into the HASH_DIN register (where ‘U’ means don’t care)
2. 0x18 is written into the HASH_STR register (the number of valid bits in the last word written into the HASH_DIN register is 24, as the original message length is 24 bits)
3. 0x10 is written into the HASH_STR register to start the message padding and digest computation. When NBLW ≠ 0x00, the message padding puts a “1” into the HASH_DIN register at the bit position defined by the NBLW value, and inserts “0”s at bit locations [31:(NBLW+1)]. When NBLW == 0x00, the message padding inserts one new word with value 0x0000 0001. Then an all zero word (0x0000 0000) is added and the message length in a two-word representation, to get a block of 16 x 32-bit words.
4. The HASH computing is performed, and the message digest is then available in the HASH_Hx registers (x = 0...4) for the SHA-1 algorithm. For example:

```

H0 = 0xA9993E36
H1 = 0x4706816A
H2 = 0xBA3E2571
H3 = 0x7850C26C
H4 = 0x9CD0D89D

```

21.3.5 Hash operation

The hash function (SHA-1, MD5) is selected when the INIT bit is written to ‘1’ in the HASH_CR register while the MODE bit is at ‘0’ in HASH_CR. The algorithm (SHA-1 or MD5) is selected at the same time (that is when the INIT bit is set) using the ALGO bit.

The message can then be sent by writing it word by word into the HASH_DIN register. When a block of 512 bits —that is 16 words— has been written, a partial digest computation starts upon writing the first data of the next block. The hash processor remains busy for 66 cycles for the SHA-1 algorithm or 50 cycles for the MD5 algorithm.

The process can then be repeated until the last word of the message. If DMA transfers are used, refer to the [Procedure where the data are loaded by DMA](#) section. Otherwise, if the message length is not an exact multiple of 512 bits, then the HASH_STR register has to be written to launch the computation of the final digest.

Once computed, the digest can be read from the HASH_H0...HASH_H4 registers (for the MD5 algorithm, HASH_H4 is not relevant).

21.3.6 HMAC operation

The HMAC algorithm is used for message authentication, by irreversibly binding the message being processed to a key chosen by the user. For HMAC specifications, refer to ‘HMAC: keyed-hashing for message authentication, H. Krawczyk, M. Bellare, R. Canetti, February 1997.

Basically, the algorithm consists of two nested hash operations:

```
HMAC(message) = Hash[ ((key | pad) XOR 0x5C)
| Hash(((key | pad) XOR 0x36) | message)]
```

where:

- pad is a sequence of zeroes needed to extend the key to the length of the underlying hash function data block (that is 512 bits for both the SHA-1 and MD5 hash algorithms)
- | represents the concatenation operator

To compute the HMAC, four different phases are required:

1. The block is initialized by writing the INIT bit to ‘1’ with the MODE bit at ‘1’ and the ALGO bit set to the value corresponding to the desired algorithm. The LKEY bit must also be set during this phase if the key being used is longer than 64 bytes (in this case, the HMAC specifications specify that the hash of the key should be used in place of the real key).
2. The key (to be used for the inner hash function) is then given to the core. This operation follows the same mechanism as the one used to send the message in the hash operation (that is, by writing into HASH_DIN and, finally, into HASH_STR).
3. Once the last word has been entered and computation has started, the hash processor elaborates the key. It is then ready to accept the message text using the same mechanism as the one used to send the message in the hash operation.
4. After the first hash round, the hash processor returns “ready” to indicate that it is ready to receive the key to be used for the outer hash function (normally, this key is the same as the one used for the inner hash function). When the last word of the key is entered and computation starts, the HMAC result is made available in the HASH_H0...HASH_H4 registers.

Note: 1 The computation latency of the HMAC primitive depends on the lengths of the keys and message. You could the HMAC as two nested underlying hash functions with the same key length (long or short).

21.3.7 Context swapping

It is possible to interrupt a hash/HMAC process to perform another processing with a higher priority, and to complete the interrupted process later on, when the higher-priority task is complete. To do so, the context of the interrupted task must be saved from the hash registers to memory, and then be restored from memory to the hash registers.

The procedures where the data flow is controlled by software or by DMA are described below.

Procedure where the data are loaded by software

The context can be saved only when no block is currently being processed. That is, you must wait for DINIS = 1 (the last block has been processed and the input FIFO is empty) or NBW ≠ 0 (the FIFO is not full and no processing is ongoing).

- Context saving:

Store the contents of the following registers into memory:

- HASH_IMR
- HASH_STR
- HASH_CR
- HASH_CSR0 to HASH_CSR50

- Context restoring:

The context can be restored when the high-priority task is complete. Please follow the order of the sequence below.

- a) Write the following registers with the values saved in memory: HASH_IMR, HASH_STR and HASH_CR
- b) Initialize the hash processor by setting the INIT bit in the HASH_CR register
- c) Write the HASH_CSR0 to HASH_CSR50 registers with the values saved in memory

You can now restart the processing from the point where it has been interrupted.

Procedure where the data are loaded by DMA

In this case it is not possible to predict if a DMA transfer is in progress or if the process is ongoing. Thus, you must stop the DMA transfers, then wait until the HASH is ready in order to interrupt the processing of a message.

- Interrupting a processing:

- Clear the DMAE bit to disable the DMA interface
- Wait until the current DMA transfer is complete (wait for DMAES = 0 in the HASH_SR register). Note that the block may or not have been totally transferred to the HASH.
- Disable the corresponding channel in the DMA controller
- Wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1

- The context saving and context restoring phases are the same as above (see [Procedure where the data are loaded by software](#)).

Reconfigure the DMA controller so that it transfers the end of the message. You can now restart the processing from the point where it was interrupted by setting the DMAE bit.

- Note:*
- 1 *If context swapping does not involve HMAC operations, the HASH_CSR38 to HASH_CSR50 registers do not have to be saved and restored.*
 - 2 *If context swapping occurs between two blocks (the last block was completely processed and the next block has not yet been pushed into the IN FIFO, NBW = 000 in the HASH_CR register), the HASH_CSR22 to HASH_CSR37 registers do not have to be saved and restored.*

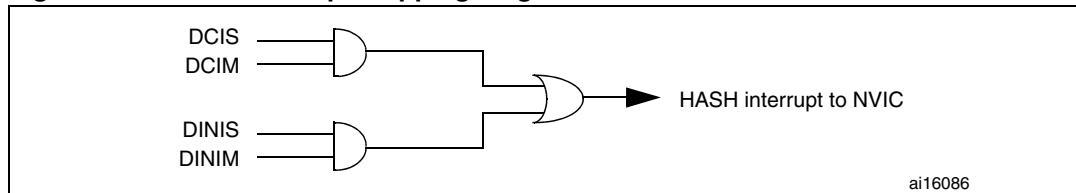
21.3.8 HASH interrupt

There are two individual maskable interrupt sources generated by the HASH processor. They are connected to the same interrupt vector.

You can enable or disable the interrupt sources individually by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit to 1 enables the interrupt.

The status of the individual interrupt sources can be read from the HASH_SR register.

Figure 207. HASH interrupt mapping diagram



21.4 HASH registers

The HASH core is associated with several control and status registers and five message digest registers.

All these registers are accessible through word accesses only, else an AHB error is generated.

21.4.1 HASH control register (HASH_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	LKEY
Reserved																rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	DINNE	NBW				ALGO	MODE	DATATYPE		DMAE	INIT	Reserved				
	r	r	r	r	r	rw	rw	rw	rw	rw	w					

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

Bit 16 **LKEY**: Long key selection

This bit selects between short key (\leq 64 bytes) or long key ($>$ 64 bytes) in HMAC mode

0: Short key (\leq 64 bytes)

1: Long key ($>$ 64 bytes)

Note: This selection is only taken into account when the INIT bit is set and MODE = 1.

Changing this bit during a computation has no effect.

Bits 15:13 Reserved, must be kept at reset value

Bit 12 **DINNE**: DIN not empty

This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is written to 1.

0: No data are present in the data input buffer

1: The input buffer contains at least one word of data

Bits 11:8 **NBW**: Number of words already pushed

This bitfield reflects the number of words in the message that have already been pushed into the IN FIFO.

NBW increments (+1) when a write access is performed to the HASH_DIN register while DINNE = 1.

It goes to 0000 when the INIT bit is written to 1 or when a digest calculation starts (DCAL written to 1 or DMA end of transfer).

- If the DMA is not used:

0000 and DINNE=0: no word has been pushed into the DIN buffer (the buffer is empty, both the HASH_DIN register and the IN FIFO are empty)

0000 and DINNE=1: 1 word has been pushed into the DIN buffer (The HASH_DIN register contains 1 word, the IN FIFO is empty)

0001: 2 words have been pushed into the DIN buffer (the HASH_DIN register and the IN FIFO contain 1 word each)

...

1111: 16 words have been pushed into the DIN buffer

- If the DMA is used, NBW is the exact number of words that have been pushed into the IN FIFO.

Bit 7 ALGO: Algorithm selection

This bit selects the SHA-1 or the MD5 algorithm:

- 0: SHA-1 algorithm selected
- 1: MD5 algorithm selected

Note: This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bit 6 MODE: Mode selection

This bit selects the HASH or HMAC mode for the selected algorithm:

- 0: Hash mode selected
- 1: HMAC mode selected. LKEY must be set if the key being used is longer than 64 bytes.

Note: This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bits 5:4 DATATYPE: Data type selection

Defines the format of the data entered into the HASH_DIN register:

- 00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.

- 01: 16-bit data, or half-word. The data written into HASH_DIN are considered as 2 half-words, and are swapped before being used by the HASH processing.

- 10: 8-bit data, or bytes. The data written into HASH_DIN are considered as 4 bytes, and are swapped before being used by the HASH processing.

- 11: bit data, or bit-string. The data written into HASH_DIN are considered as 32 bits (1st bit of the sting at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 DMAE: DMA enable

- 0: DMA transfers disabled

- 1: DMA transfers enabled. A DMA request is sent as soon as the HASH core is ready to receive data.

Note: 1: This bit is cleared by hardware when the DMA asserts the DMA terminal count signal (while transferring the last data of the message). This bit is not cleared when the INIT bit is written to 1.

2: If this bit is written to 0 while a DMA transfer has already been requested to the DMA, DMAE is cleared but the current transfer is not aborted. Instead, the DMA interface remains internally enabled until the transfer is complete or INIT is written to 1.

Bit 2 INIT: Initialize message digest calculation

Writing this bit to 1 resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.

Writing this bit to 0 has no effect.

Reading this bit always return 0.

Bit 1:0 Reserved, must be kept at reset value

21.4.2 HASH data input register (HASH_DIN)

Address offset: 0x04

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. It is used to enter the message by blocks of 512 bits. When the HASH_DIN register is written to, the value presented on the AHB databus is ‘pushed’ into the HASH core and the register takes the new value presented on the AHB databus. The DATATYPE bits must previously have been configured in the HASH_CR register to get a correct message representation.

When a block of 16 words has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing new data into the HASH_DIN register (the first word of the next block) if the DMA is not used (intermediate digest calculation)
- automatically if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched:

- by writing the DCAL bit to 1 in the HASH_STR register (final digest calculation)
- automatically if the DMA is used

When a digest calculation (intermediate or final) is in progress, any new write access to the HASH_DIN register is extended (by wait-state insertion on the AHB bus) until the HASH calculation completes.

When the HASH_DIN register is read, the last word written in this location is accessed (zero after reset).

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

Bit 31:0 **DATAIN**: Data input

Read = returns the current register content.

Write = the current register content is pushed into the IN FIFO, and the register takes the new value presented on the AHB databus.

21.4.3 HASH start register (HASH_STR)

Address offset: 0x08

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written into the HASH_DIN register)
- It is used to start the processing of the last block in the message by writing the DCAL bit to 1

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

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

Bit 8 **DCAL**: Digest calculation

Writing this bit to 1 starts the message padding, using the previously written value of NBLW, and starts the calculation of the final message digest with all data words written to the IN FIFO since the INIT bit was last written to 1.

Reading this bit returns 0.

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

Bits 4:0 **NBLW**: Number of valid bits in the last word of the message

When these bits are written and DCAL is at '0', they take the value on the AHB databus:

0x00: All 32 bits of the last data written in the HASH_DIN register are valid

0x01: Only bit [0] of the last data written in the HASH_DIN register is valid

0x02: Only bits [1:0] of the last data written in the HASH_DIN register are valid

0x03: Only bits [2:0] of the last data written in the HASH_DIN register are valid

...

0x1F: Only bits [30:0] of the last data written in the HASH_DIN register are valid

When these bits are written and DCAL is at '1', the bitfield is not changed.

Reading them returns the last value written to NBLW.

Note: These bits must be configured before setting the DCAL bit, else they are not taken into account. Especially, it is not possible to configure NBLW and set DCAL at the same time.

21.4.4 HASH digest registers (HASH_HR0...4)

Address offset: 0x0C to 0x1C

Reset value: 0x0000 0000

These registers contain the message digest result named as H0, H1, H2, H3 and H4, respectively, in the HASH algorithm description, and as A, B, C and D, respectively, in the MD5 algorithm description (note that in this case, the HASH_H4 register is not used, and is read as zero).

If a read access to one of these registers occurs while the HASH core is calculating an intermediate digest or a final message digest (that is when the DCAL bit has been written to 1), then the access on the AHB bus is extended until the completion of the HASH calculation.

HASH_HR0 (address offset: 0x0C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H0															
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
H0															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR1 (address offset: 0x10)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H1															
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
H1															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR2 (address offset: 0x14)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H2															
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
H2															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR3 (address offset: 0x18)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H3															
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
H3															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

HASH_HR4 (address offset: 0x1C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H4															
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
H4															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Note: When starting a digest computation for a new bit stream (by writing the INIT bit to 1), these registers assume their reset values.

21.4.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value

Bit 1 **DCIE:** Digest calculation completion interrupt enable

- 0: Digest calculation completion interrupt disabled
- 1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable

- 0: Data input interrupt disabled
- 1: Data input interrupt enabled

21.4.6 HASH status register (HASH_SR)

Address offset: 0x24

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
												BUSY	DMAS	DCIS	DINIS
												r	r	rc_w0	rc_w0

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

Bit 3 **BUSY**: Busy bit

- 0: No block is currently being processed
- 1: The hash core is processing a block of data

Bit 2 **DMAS**: DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE=0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

- 0: DMA interface is disabled (DMAE=0) and no transfer is ongoing
- 1: DMA interface is enabled (DMAE=1) or a transfer is ongoing

Bit 1 **DCIS**: Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by writing the INIT bit to 1 in the HASH_CR register.

- 0: No digest available in the HASH_Hx registers
- 1: Digest calculation complete, a digest is available in the HASH_Hx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS**: Data input interrupt status

This bit is set by hardware when the input buffer is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.

- 0: Less than 16 locations are free in the input buffer
- 1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

21.4.7 HASH context swap registers (HASH_CSR0...50)

Address offset: 0x0F8 to 0x1C0

Reset value: 0x0000 0000

These registers contain the complete internal register states of the hash processor, and are useful when a context swap has to be done because a high-priority task has to use the hash processor while it is already in use by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved somewhere in the system memory space. Then the hash processor can be used by the preemptive task, and when hash computation is finished, the saved context can be read from memory and written back into these HASH_CSRx registers.

HASH_CSRx (address offset: 0x0F8 to 0x1C0)

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

21.4.8 HASH register map

Table 75 gives the summary HASH register map and reset values.

Table 75. HASH register map and reset values

Offset	Register name reset value	Register size																																	
		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			
0x00	HASH_CR	Reserved														LKEY	Reserved	NBW				ALGO	MODE	DATATYPE	DMAE	INIT	Reserved	0							
	Reset value															0		0	0	0	0	0	0	0	0	0	0	0							
0x04	HASH_DIN	DATAIN																																	
	Reset value	0	0	0	0	0	0	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	HASH_STR	Reserved														DCAL	NBLW				Reserved	NBLW				0	0	0	0	0	0				
	Reset value																		0																
0x0C	HASH_HR0	H0																																	
	Reset value	0	0	0	0	0	0	0	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	HASH_HR1	H1																																	
	Reset value	0	0	0	0	0	0	0	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	HASH_HR2	H2																																	
	Reset value	0	0	0	0	0	0	0	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	HASH_HR3	H3																																	
	Reset value	0	0	0	0	0	0	0	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	HASH_HR4	H4																																	
	Reset value	0	0	0	0	0	0	0	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	HASH_IMR	Reserved																																	
	Reset value	0	0	0	0	0	0	0	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	HASH_SR	Reserved																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xF8	HASH_CSR0	CSR0																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x1C0	HASH_CSR50	CSR50																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

22 Real-time clock (RTC)

22.1 Introduction

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar, two programmable alarm interrupts, and a periodic programmable wakeup flag with interrupt capability. The RTC also includes an automatic wakeup unit to manage low power modes.

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 power-on 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).

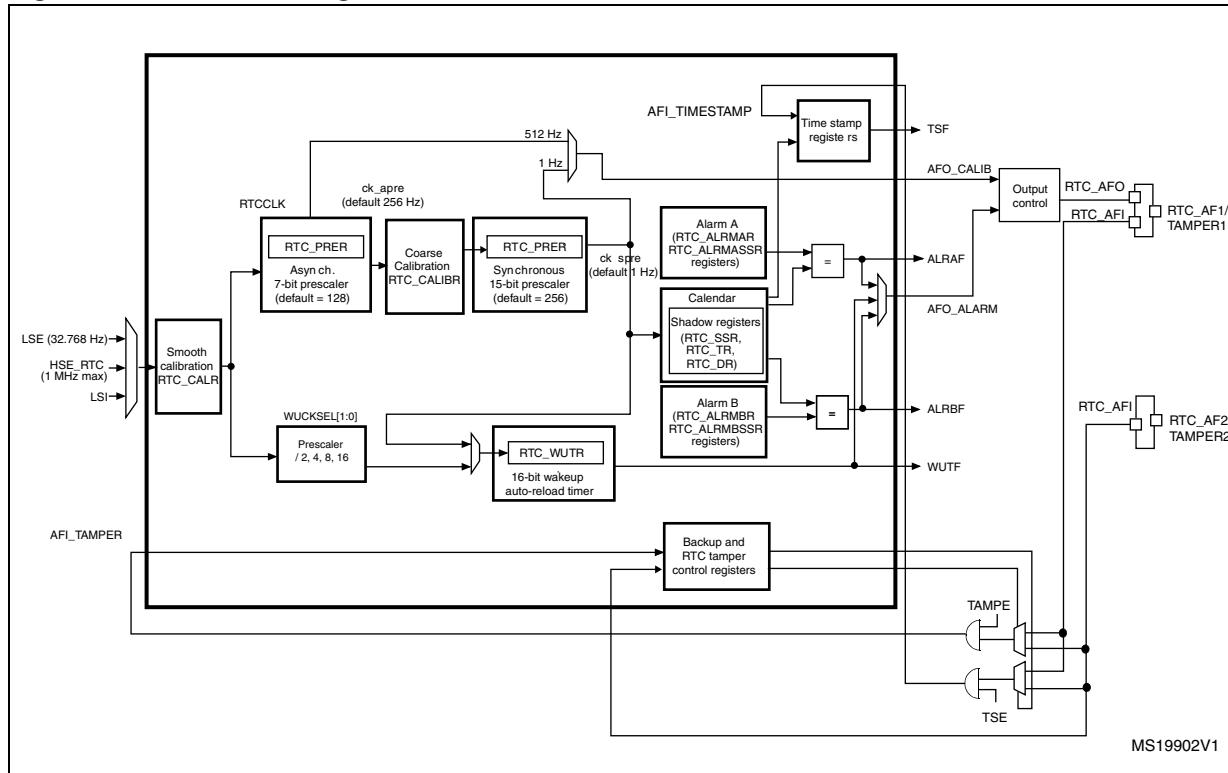
22.2 RTC main features

The RTC unit main features are the following (see [Figure 208: 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.
- Two programmable alarms with interrupt function. The alarms 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.
- Maskable interrupts/events:
 - Alarm A
 - Alarm B
 - Wakeup interrupt
 - Timestamp
 - Tamper detection
- Digital calibration circuit (periodic counter correction)
 - 5 ppm accuracy
 - 0.95 ppm accuracy, obtained in a calibration window of several seconds
- Timestamp function for event saving (1 event)
- Tamper detection:
 - 2 tamper events with configurable filter and internal pull-up.
- 20 backup registers (80 bytes). The backup registers are reset when a tamper detection event occurs.
- RTC alternate function outputs (RTC_AFO):
 - AFO_CALIB: 512 Hz or 1Hz clock output (with an LSE frequency of 32.768 kHz). It is routed to the device RTC_AF1 pin.
 - AFO_ALARM: Alarm A or Alarm B or wakeup (only one can be selected). It is routed to the device RTC_AF1 pin.
- RTC alternate function inputs (RTC_AFI):
 - AFI_TAMPER1: tamper event detection. It is routed to the device RTC_AF1 and RTC_AF2 pins.
 - AFI_TAMPER2 : tamper2 event detection. It is routed to the device RTC_TAMPER2 pin.
 - AFI_TIMESTAMP: timestamp event detection. It is routed to the device RTC_AF1 and RTC_AF2 pins.

Note: Refer to [Section 6.3.15: Selection of RTC_AF1 and RTC_AF2 alternate functions](#) for more details on how to select RTC alternate functions (RTC_AF1 and RTC_AF2).

Figure 208. RTC block diagram



1. On STM32F40x and STM32F41x devices, the RTC_AF1 and RTC_AF2 alternate functions are connected to PC13 and PI8, respectively.

22.3 RTC functional description

22.3.1 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to [Section 5: Reset and clock control \(RCC\)](#).

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see [Figure 208: 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 1 MHz.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREVID_A + 1)}$$

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 22.3.4: Periodic auto-wakeup](#) for details).

22.3.2 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK1 (APB1 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 22.6.4](#)). 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.

22.3.3 Programmable alarms

The RTC unit provides two programmable alarms, Alarm A and Alarm B.

The programmable alarm functions are enabled through the ALRAE and ALRBE bits in the RTC_CR register. The ALRAF and ALRBF flags are set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR, respectively. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR and RTC_ALRMBR registers, and through the MASKSSx bits of the RTC_ALRMASSR and RTC_ALRMBSSR registers. The alarm interrupts are enabled through the ALRAIE and ALRBE bits in the RTC_CR register.

Alarm A and Alarm B (if enabled by bits OSEL[0:1] in RTC_CR register) can be routed to the AFO_ALARM output. AFO_ALARM polarity can be configured through bit POL the RTC_CR register.

Caution: If the seconds field is selected (MSK0 bit reset in RTC_ALRMAR or RTC_ALRMBR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

22.3.4 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 µs to 32 s, with a resolution down to 61µs.
- 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 542](#)), 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 AFO_ALARM output provided it has been enabled through bits OSEL[0:1] of RTC_CR register. AFO_ALARM 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.

22.3.5 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

RTC register write protection

After power-on reset, 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_ISR[13:8], RTC_TAFCR, and RTC_BKPxR.

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 first the synchronous prescaler factor in RTC_PRER register, and then program the asynchronous prescaler factor. Even if only one of the two fields needs to be changed, 2 separate write accesses must be performed to the TC_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:*
- 1 *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 power-on reset default value (0x00).*
 - 2 *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 alarm (Alarm A or Alarm B):

1. Clear ALRAE or ALRBE in RTC_CR to disable Alarm A or Alarm B.
2. Poll ALRAWF or ALRBWF in RTC_ISR until it is set to make sure the access to alarm registers is allowed. This takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the Alarm A or Alarm B registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRMBSSR/RTC_ALRMBR).
4. Set ALRAE or ALRBE in the RTC_CR register to enable Alarm A or Alarm B 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.

22.3.6 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_{PCLK1}) must be equal to or greater than seven times the f_{RTCCLK} RTC clock frequency. 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_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.

- Note:*
- 1 *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.*
 - 2 *After an initialization (refer to [Calendar initialization and configuration on page 541](#)): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.*
 - 3 *After synchronization (refer to [Section 22.3.8: RTC synchronization](#)): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.*

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low power modes (STOP or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

- Note:*
- While BYPSHAD=1, instructions which read the calendar registers require one extra APB cycle to complete.*

22.3.7 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and 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 power-on reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration registers (RTC_CALIBR or RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDDR), 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_ALRMASSR/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 power-on reset one. When a power-on reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

22.3.8 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’s 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 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0xFFFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler’s output at 1 Hz. In this way, the frequency of the asynchronous prescaler’s output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of $1 / (\text{PREDIV_S} + 1)$ seconds. The shift operation consists of adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this will delay the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this will advance the clock.

Caution: Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow will occur.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON=1.

22.3.9 RTC reference clock detection

The reference clock (at 50 Hz or 60 Hz) should have a higher precision than the 32.768 kHz LSE clock. When the reference clock detection is enabled (REFCKON bit of RTC_CR set to 1), it is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest reference 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.

If the reference clock halts, the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a detection window centered on the ck_spre edge.

When the reference clock detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREDIV_S = 0x00FF

Note: The reference clock detection is not available in Standby mode.

Caution: The reference clock detection feature cannot be used in conjunction with the coarse digital calibration: RTC_CALIBR must be kept at 0x0000 0000 when REFCKON=1.

22.3.10 RTC coarse digital calibration

Two digital calibration methods are available: coarse and smooth calibration. To perform coarse calibration refer to [Section 22.6.7: RTC calibration register \(RTC_CALIBR\)](#).

The two calibration methods are not intended to be used together, the application must select one of the two methods. Coarse calibration is provided for compatibility reasons. To perform smooth calibration refer to [Section 22.3.11: RTC smooth digital calibration](#) and the [Section 22.6.16: RTC calibration register \(RTC_CALR\)](#)

The coarse digital calibration can be used to compensate crystal inaccuracy by adding (positive calibration) or masking (negative calibration) clock cycles at the output of the asynchronous prescaler (ck_apre).

Positive and negative calibration are selected by setting the DCS bit in RTC_CALIBR register to '0' and '1', respectively.

When positive calibration is enabled (DCS = '0'), 2 ck_apre cycles are added every minute (around 15360 ck_apre cycles) for 2xDC minutes. This causes the calendar to be updated sooner, thereby adjusting the effective RTC frequency to be a bit higher.

When negative calibration is enabled (DCS = '1'), 1 ck_apre cycle is removed every minute (around 15360 ck_apre cycles) for 2xDC minutes. This causes the calendar to be updated later, thereby adjusting the effective RTC frequency to be a bit lower.

DC is configured through bits DC[4:0] of RTC_CALIBR register. This number ranges from 0 to 31 corresponding to a time interval (2xDC) ranging from 0 to 62.

The coarse digital calibration can be configured only in initialization mode, and starts when the INIT bit is cleared. The full calibration cycle lasts 64 minutes. The first 2xDC minutes of the 64 -minute cycle are modified as just described.

Negative calibration can be performed with a resolution of about 2 ppm while positive calibration can be performed with a resolution of about 4 ppm. The maximum calibration ranges from -63 ppm to 126 ppm.

The calibration can be performed either on the LSE or on the HSE clock.

Caution: Digital calibration may not work correctly if PREDIV_A < 6.

Case of RTCCLK=32.768 kHz and PREDIV_A+1=128

The following description assumes that ck_apre frequency is 256 Hz obtained with an LSE clock nominal frequency of 32.768 kHz, and PREDIV_A set to 127 (default value).

The ck_spred clock frequency is only modified during the first 2xDC minutes of the 64-minute cycle. For example, when DC equals 1, only the first 2 minutes are modified. This means that the first 2xDC minutes of each 64-minute cycle have, once per minute, one second either shortened by 256 or lengthened by 128 RTCCLK cycles, given that each ck_apre cycle represents 128 RTCCLK cycles (with PREDIV_A+1=128).

Therefore each calibration step has the effect of adding 512 or subtracting 256 oscillator cycles for every 125829120 RTCCLK cycles (64min x 60 s/min x 32768 cycles/s). This is equivalent to +4.069 ppm or -2.035 ppm per calibration step. As a result, the calibration

resolution is +10.5 or -5.27 seconds per month, and the total calibration ranges from +5.45 to -2.72 minutes per month.

In order to measure the clock deviation, a 512 Hz clock is output for calibration. Refer to [Section 22.3.14: Calibration clock output](#).

22.3.11 RTC smooth digital calibration

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.

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 SMC[2] to 1 causes four additional cycles to be masked
- and so on up to SMC[8] set to 1 which causes 256 clocks to be masked.

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 performed 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 stucked 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.

22.3.12 Timestamp function

Timestamp is enabled by setting the TSE bit of RTC_CR register to 1.

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the pin to which the TIMESTAMP alternate function

is mapped. When a timestamp event occurs, the timestamp 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 timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

- Note:*
- 1 *TSF is set 2 ck_apre cycles after the timestamp event occurs due to synchronization process.*
 - 2 *There is no delay in the setting of TSOVF. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.*

If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write '0' into TSF bit unless it has already read it to '1'. Optionally, a tamper event can cause a timestamp to be recorded. See the description of the TAMPTS control bit in [Section 22.6.17: RTC tamper and alternate function configuration register \(RTC_TAFCR\)](#). If the timestamp event is on the same pin than a tamper event configured in filtered mode (TAMPFLT set to a non-zero value), the timestamp on tamper detection event mode must be selected by setting TAMPTS='1' in RTC_TAFCR register.

TIMESTAMP alternate function

The TIMESTAMP alternate function can be mapped to either RTC_AF1 or RTC_AF2 depending on the value of the TSINSEL bit in the RTC_TAFCR register (see [Section 22.6.17 on page 569](#)). Mapping the timestamp event on RTC_AF2 is not allowed if RTC_AF1 is used as TAMPER in filtered mode (TAMPFLT set to a non-zero value).

22.3.13 Tamper detection

Two tamper detection inputs are available. They can be configured either for edge detection, or for level detection with filtering.

RTC backup registers

The backup registers (RTC_BKPxR) are twenty 32-bit registers for storing 80 bytes of user application data. They are implemented in the backup domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off. They are not reset by system reset, power-on reset, or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs (see [Section 22.6.20: RTC backup registers \(RTC_BKPxR\)](#) and [Tamper detection initialization on page 548](#)).

Tamper detection initialization

Each tamper detection input is associated with a flag TAMP1F/TAMP2F in the RTC_ISR2 register. Each input can be enabled by setting the corresponding TAMP1E/TAMP2E bits to 1 in the RTC_TAFCR register.

A tamper detection event resets all backup registers (RTC_BKPxR).

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 (TAMP1F, TAMP2F) is set at the same time that TSF or TSOVF is set.

Edge detection on tamper inputs

If the TAMPFLT bits are “00”, the TAMPER pins generate tamper detection events(AFI_TAMPER[2:1]) when either a rising edge is observed or an falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMPER 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 TAMPxE in order to detect a tamper detection event in case it occurs before the TAMPERx pin is enabled.

- When TAMPxTRG = 0: if the TAMPERx alternate function is already high before tamper detection is enabled (TAMPxE bit set to 1), a tamper event is detected as soon as TAMPERx is enabled, even if there was no rising edge on TAMPERx after TAMPxE was set.
- When TAMPxTRG = 1: if the TAMPERx alternate function is already low before tamper detection is enabled, a tamper event is detected as soon as TAMPERx is enabled (even if there was no falling edge on TAMPERx after TAMPxE was set).

After a tamper event has been detected and cleared, the TAMPERx 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 TAMPERx value still indicates a tamper detection. This is equivalent to a level detection on the TAMPERx alternate function.

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 TAMPER alternate function is mapped should be externally tied to the correct level.*

Level detection with filtering on tamper 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 (TAMP1TRG/TAMP2TRG).

The TAMPER inputs are pre-charged 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 tamper inputs.

The tradeoff 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.*

TAMPER alternate function detection

The TAMPER1 alternate function can be mapped either to RTC_AF1(PC13) or RTC_AF2 (P18) depending on the value of TAMP1INSEL bit in RTC_TAFCR register (see [Section 22.6.17: RTC tamper and alternate function configuration register \(RTC_TAFCR\)](#)).

TAMPE bit must be cleared when TAMP1INSEL is modified to avoid unwanted setting of TAMPF.

The TAMPER 2 alternate functions is mapped to RTC_TAMPER2 pin.

22.3.14 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the RTC_CALIB device output. If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the RTC_CALIB frequency is $f_{RTCCLK}/64$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz.

The RTC_CALIB output is not impacted by the calibration value programmed in RTC_CALIBR register. The RTC_CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

If 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 $f_{RTCCLK}/(256 * (PREDIV_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = 0x7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Calibration alternate function output

When the COE bit in the RTC_CR register is set to 1, the calibration alternate function (AFO_CALIB) is enabled on RTC_AF1.

22.3.15 Alarm output

Three functions can be selected on Alarm output: ALRAF, ALRBF and WUTF. These functions reflect the contents of the corresponding flags in the RTC_ISR register.

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm alternate function output (AFO_ALARM) in RTC_AF1, and to select the function which is output on AFO_ALARM.

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

AFO_ALARM can be configured in output open drain or output push-pull using the control bit ALARMOUTTYPE in the RTC_TAFCR register.

- Note:*
- 1 Once AFO_ALARM is enabled, it has priority over AFO_CALIB (COE bit is don't care on RTC_AF1).
 - 2 When AFO_CALIB or AFO_ALARM is selected, RTC_AF1 is automatically configured in output alternate function.

22.4 RTC and low power modes

Table 76. 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 time stamp 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 time stamp event, and RTC Wakeup cause the device to exit the Standby mode.

22.5 RTC interrupts

All RTC interrupts are connected to the EXTI controller.

To enable the RTC Alarm interrupt, the following sequence is required:

1. Configure and enable the EXTI Line 17 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 (Alarm A or Alarm B).

To enable the RTC Wakeup interrupt, the following sequence is required:

1. Configure and enable the EXTI Line 22 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 generate the RTC wakeup timer event.

To enable the RTC Tamper interrupt, the following sequence is required:

1. Configure and enable the EXTI Line 21 in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the 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 21 in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the TAMP_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC timestamp event.

Table 77. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
Alarm A	ALRAF	ALRAIE	yes	yes ⁽¹⁾	yes ⁽¹⁾
Alarm B	ALRBF	ALRBIE	yes	yes ⁽¹⁾	yes ⁽¹⁾
Wakeup	WUTF	WUTIE	yes	yes ⁽¹⁾	yes ⁽¹⁾
TimeStamp	TSF	TSIE	yes	yes ⁽¹⁾	yes ⁽¹⁾
Tamper1 detection	TAMP1F	TAMPIE	yes	yes ⁽¹⁾	yes ⁽¹⁾
Tamper2 detection ⁽²⁾	TAMP2F	TAMPIE	yes	yes ⁽¹⁾	yes ⁽¹⁾

1. Wakeup from STOP and Standby modes is possible only when the RTC clock source is LSE or LSI.

2. If RTC_TAMPER2 pin is present. Refer to device datasheet pinout.

22.6 RTC registers

Refer to [Section 1.1](#) of the reference manual for a list of abbreviations used in register descriptions.

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

22.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to [Calendar initialization and configuration on page 541](#) and [Reading the calendar on page 542](#).

Address offset: 0x00

Power-on reset value: 0x0000 0000

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

Bits 31-24 Reserved

Bit 23 Reserved, always read as 0.

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 16:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, always read as 0.

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, always read as 0.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bit 3:0 **SU[3:0]**: Second units in BCD format

Note: *This register is write protected. The write access procedure is described in [RTC register write protection on page 541](#).*

22.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to [Calendar initialization and configuration on page 541](#) and [Reading the calendar on page 542](#).

Address offset: 0x04

Power-on reset value: 0x0000 2101

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

Bits 31-24 Reserved

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, always read as 0.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: This register is write protected. The write access procedure is described in [RTC register write protection on page 541](#).

22.6.3 RTC control register (RTC_CR)

Address offset: 0x08

Power-on value: 0x0000 0000

Reset value: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	w	w		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	DCE	FMT	BYPS HAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, always read as 0.

Bit 23 **COE**: Calibration output enable

This bit enables the AFO_CALIB RTC 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 AFO_ALARM RTC 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 AFO_ALARM RTC 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 22.3.14: Calibration clock output](#)

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect

1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable

0: Timestamp Interrupt disable

1: Timestamp Interrupt enable

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**: Time stamp enable

0: Time stamp disable

1: Time stamp 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 **DCE**: Coarse digital calibration enable

0: Digital calibration disabled

1: Digital calibration enabled

PREDIV_A must be 6 or greater

Bit 6 **FMT**: Hour format

0: 24 hour/day format

1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to '1'.

Bit 4 **REFCKON**: Reference clock detection enable (50 or 60 Hz)

0: Reference clock detection disabled

1: Reference clock detection enabled

Note: PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge

0: TIMESTAMP rising edge generates a timestamp event

1: TIMESTAMP falling edge generates a timestamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting

Bits 2:0 **WUCKSEL[2:0]**: 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:**
- 1 *WUT = Wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].*
 - 2 *Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC_ISR/INITF = 1).*
 - 3 *Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ISR WUTWF bit = 1.*
 - 4 *It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.*
 - 5 *ADD1H and SUB1H changes are effective in the next second.*
 - 6 *This register is write protected. The write access procedure is described in [RTC register write protection on page 541](#).*

22.6.4 RTC initialization and status register (RTC_ISR)

Address offset: 0x0C

Reset value: 0x0000 0007

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	RECALPF
Reserved																r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF	
	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	rc_w0	r	r	r	

Bits 31:17 Reserved

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPFstatus 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 [Section : Re-calibration on-the-fly](#).

Bit 15 Reserved, always read as 0.

Bit 14 **TAMP2F**: TAMPER2 detection flag

This flag is set by hardware when a tamper detection event is detected on tamper input 2. It is cleared by software writing 0.

Bit 13 **TAMP1F: Tamper detection flag**

This flag is set by hardware when a tamper detection event is detected.
It is cleared by software writing 0.

Bit 12 **TSOVF: Timestamp overflow flag**

This flag is set by hardware when a timestamp 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 timestamp event occurs immediately before the TSF bit is cleared.

Bit 11 **TSF: Timestamp flag**

This flag is set by hardware when a timestamp event occurs.
This flag is cleared by software by writing 0.

Bit 10 **WUTF: Wakeup timer flag**

This flag is set by hardware when the wakeup auto-reload counter reaches 0.
This flag is cleared by software by writing 0.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 9 **ALRBF: Alarm B flag**

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm B register (RTC_ALRMBR).
This flag is cleared by software by writing 0.

Bit 8 **ALRAF: Alarm A flag**

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm A register (RTC_ALRMAR).
This flag is cleared by software by writing 0.

Bit 7 **INIT: Initialization mode**

0: Free running mode
1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF: Initialization flag**

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.
0: Calendar registers update is not allowed
1: Calendar registers update is allowed.

Bit 5 **RSF: Registers synchronization flag**

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow register mode (BYPSHAD=1). This bit can also be cleared by software.
0: Calendar shadow registers not yet synchronized
1: Calendar shadow registers synchronized

Bit 4 **INITS: Initialization status flag**

This bit is set by hardware when the calendar year field is different from 0 (power-on reset state).
0: Calendar has not been initialized
1: Calendar has been initialized

Bit 3 **SHPF**:Shift operation pending

- 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. It is cleared by hardware when the corresponding shift operation has been executed. Writing to SHPF 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:*
- 1 *The ALRAF, ALRBF, WUTF and TSF bits are cleared 2 APB clock cycles after programming them to 0.*
 - 2 *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 541](#).*

22.6.5 RTC prescaler register (RTC_PRER)

Address offset: 0x10

Power-on reset value: 0x007F 00FF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														PREDIV_A[6:0]	
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	rw

Bits 31:24 Reserved

Bit 23 Reserved, always read as 0.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:

$$\text{ck_apre frequency} = \text{RTCCLK frequency}/(\text{PREDIV_A}+1)$$

Note: PREDIV_A [6:0]= 000000 is a prohibited value.

Bit 15 Reserved, always read as 0.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:

$$\text{ck_spre frequency} = \text{ck_apre frequency}/(\text{PREDIV_S}+1)$$

- Note:*
- 1 *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 541](#)*
 - 2 *This register is write protected. The write access procedure is described in [RTC register write protection on page 541](#).*

22.6.6 RTC wakeup timer register (RTC_WUTR)

Address offset: 0x14

Power-on reset value: 0x0000 FFFF

System reset: not affected

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
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

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.

Note: 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.

- Note:*
- 1 *This register can be written only when WUTWF is set to 1 in RTC_ISR.*
 - 2 *This register is write protected. The write access procedure is described in [RTC register write protection on page 541](#).*

22.6.7 RTC calibration register (RTC_CALIBR)

Address offset: 0x18

Power-on reset value: 0x0000 0000

System reset: not affected

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

Bits 31:8 Reserved

Bit 7 **DCS**: Digital calibration sign

- 0: Positive calibration: calendar update frequency is increased
- 1: Negative calibration: calendar update frequency is decreased

Bits 6:5 Reserved, always read as 0.

Bits 4:0 **DC[4:0]**: Digital calibration

DCS = 0 (positive calibration)

00000: +0 ppm

00001: +4 ppm

00010: +8 ppm

..

11111: +126 ppm

DCS = 1 (negative calibration)

00000: -0 ppm

00001: -2 ppm

00010: -4 ppm

..

11111: -63 ppm

Note: 1 This register can be written in initialization mode only (RTC_ISR/INITF = '1').

2 This register is write protected. The write access procedure is described in [RTC register write protection on page 541](#).

22.6.8 RTC alarm A register (RTC_ALRMAR)

Address offset: 0x1C

Power-on reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	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.

- Note:*
- 1 *This register can be written only when ALRAWF is set to 1 in RTC_ISR, or in initialization mode.*
 - 2 *This register is write protected. The write access procedure is described in [RTC register write protection on page 541](#).*

22.6.9 RTC alarm B register (RTC_ALRMBR)

Address offset: 0x20

Power-on reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	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

- Note:*
- 1 This register can be written only when ALRBWF is set to 1 in RTC_ISR, or in initialization mode.
 - 2 This register is write protected. The write access procedure is described in [RTC register write protection on page 541](#).

22.6.10 RTC sub second register (RTC_SSR)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bits31:16 Reserved

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value in the synchronous prescaler's counter. The fraction of a second is given by the formula below:

$$\text{Second fraction} = (\text{PREDIV_S} - \text{SS}) / (\text{PREDIV_S} + 1)$$

Note: SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

22.6.11 RTC shift control register (RTC_SHIFTR)

Address offset: 0x2C

Reset value: 0x0000 0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Reserved															
w	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Rsvd	SUBFS[14:0]															
r	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_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 31:15 Reserved

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's counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

$$\text{Delay (seconds)} = \text{SUBFS} / (\text{PREDIV_S} + 1)$$

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by :

$$\text{Advance (seconds)} = (1 - (\text{SUBFS} / (\text{PREDIV_S} + 1)))$$

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF=1 to be sure that the shadow registers have been updated with the shifted time.

Refer to [Section 22.3.8: RTC synchronization](#).

Note: This register is write protected. The write access procedure is described in [RTC register write protection on page 541](#)

22.6.12 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										KEY					
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:8 Reserved, always read as 0.

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.

22.6.13 RTC time stamp time register (RTC_TSTR)

Address offset: 0x30

Power-on value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved										PM	HT[1:0]		HU[3:0]			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved										Reserv-ed	ST[2:0]		SU[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:23 Reserved, always read as 0.

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, always read as 0.

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, always read as 0.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

Note: 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.

22.6.14 RTC time stamp date register (RTC_TSDDR)

Address offset: 0x34

Power-on value: 0x0000 0000

System reset: not affected

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
WDU[1:0]				MT	MU[3:0]				Reserved	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r	r		r	r	r	r	r	r

Bits 31:16 Reserved, always read as 0.

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, always read as 0.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bit 3:0 **DU[3:0]**: Date units in BCD format

Note: 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.

22.6.15 RTC timestamp sub second register (RTC_TSSSR)

Address offset: 0x38

Reset value: 0x0000 0000

System reset: not affected

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

Bits 31:16 Reserved

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value of the synchronous prescaler's counter when the timestamp event occurred.

Note: 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.

22.6.16 RTC calibration register (RTC_CALR)

Address offset: 0x3C

Power-on value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
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
CALP	CALW8	CALW16	Reserved				CALM[8:0]								
rw	rw	rw	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:16 Reserved

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: $(512 * \text{CALP}) - \text{CALM}$.

Refer to [Section 22.3.11: 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.

CALM[1:0] are stucked at "00" when CALW8='1'.

Refer to [Section 22.3.11: 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 stucked at '0' when CALW16='1'.

Refer to [Section 22.3.11: RTC smooth digital calibration](#).

Bits 12:9 Reserved

Bits 8:0 **CALM[8:0]:** Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP.

See [Section 22.3.11: RTC smooth digital calibration on page 546](#).

Note: This register is write protected. The write access procedure is described in [RTC register write protection on page 541](#)

22.6.17 RTC tamper and alternate function configuration register (RTC_TAFCR)

Address offset: 0x40

Power-on value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved														ALARMOUT TYPE	TSIN SEL	TAMP1 INSEL
														rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TAMP- PUDIS	TAMP- PRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPT S	Reserved	TAMP2 -TRG		TAMP2 E	TAMPIE	TAMP1 TRG	TAMP1 E	
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	

Bit 31:19 Reserved. Always read as 0.

Bit 18 **ALARMOUTTYPE**: AFO_ALARM output type

- 0: ALARM_AF0 is a push-pull output
- 1: ALARM_AF0 is an open-drain output

Bit 17 **TSINSEL**: TIMESTAMP mapping

- 0: RTC_AF1 used as TIMESTAMP
- 1: RTC_AF2 used as TIMESTAMP

Bit 16 **TAMP1INSEL**: TAMPER1 mapping

- 0: RTC_AF1 used as TAMPER
- 1: RTC_AF2 used as TAMPER

Note: TAMP1E must be reset when TAMP1INSEL is changed to avoid unwanted setting of TAMP1F.

Bit :3 Reserved. Always read as 0.

Bit 15 **TAMPPUDIS**: TAMPER pull-up disable

This bit determines if each of the tamper pins are pre-charged before each sample.

- 0: Precharge tamper pins before sampling (enable internal pull-up)

Note: 1: Disable precharge of tamper pins

Bits 14:13 **TAMPPRCH[1:0]**: Tamper precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the tamper inputs.

- 0x0: 1 RTCCLK cycle
- 0x1: 2 RTCCLK cycles
- 0x2: 4 RTCCLK cycles
- 0x3: 8 RTCCLK cycles

Bits 12:11 **TAMPFLT[1:0]**: Tamper filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) necessary to activate a Tamper event. TAMPFLT is valid for each of the tamper inputs.

0x0: Tamper is activated on edge of tamper input transitions to the active level (no internal pull-up on tamper input).

0x1: Tamper is activated after 2 consecutive samples at the active level.

0x2: Tamper is activated after 4 consecutive samples at the active level.

0x3: Tamper 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 tamper 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 6:5 Reserved. Always read as 0.

Bit 4 **TAMP2TRG**: Active level for tamper 2

if TAMPFLT != 00 :

0: TAMPER2 staying low triggers a tamper detection event.

1: TAMPER2 staying high triggers a tamper detection event.

if TAMPFLT = 00:

0: TAMPER2 rising edge triggers a tamper detection event.

1: TAMPER2 falling edge triggers a tamper detection event.

Bit 3 **TAMP2E**: Tamper 2 detection enable

0: Tamper 2 detection disabled

1: Tamper 2 detection enabled

Bit 2 **TAMPIE**: Tamper interrupt enable

0: Tamper interrupt disabled

1: Tamper interrupt enabled

Bit 1 **TAMP1TRG**: Active level for tamper 1

if TAMPFLT != 00

0: TAMPER1 staying low triggers a tamper detection event.

1: TAMPER1 staying high triggers a tamper detection event.

if TAMPFLT = 00:

0: TAMPER1 rising edge triggers a tamper detection event.

1: TAMPER1 falling edge triggers a tamper detection event.

Caution: When TAMPFLT = 0, TAMP1E must be reset when TAMP1TRG is changed to avoid spuriously setting TAMP1F.

Bit 0 **TAMP1E**: Tamper 1 detection enable
 0: Tamper 1 detection disabled
 1: Tamper 1 detection enabled

22.6.18 RTC alarm A sub second register (RTC_ALRMASSR)

Address offset: 0x44

Power-on value: 0x0000 0000

System reset: not affected

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

Bit 31:28 Reserved

Bit 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.

Bit 23:15 Reserved

Bit 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler's counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

Note: *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 541](#)

22.6.19 RTC alarm B sub second register (RTC_ALRMBSSR)

Address offset: 0x48

Power-on value: 0x0000 0000

System reset: not affected

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

Bit 31:28 Reserved

Bit 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.

Bit 23:15 Reserved

Bit 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler's counter to determine if Alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

Note: *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](#)

22.6.20 RTC backup registers (RTC_BKPxR)

Address offset: 0x50 to 0x9C

Power-on 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.

22.6.21 RTC register map

Table 78. RTC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
0x00	RTC_TR	Reserved				PM	HT [1:0]	HU[3:0]			Reserved	MNT[2:0]			MNU[3:0]				
		0	0	0	0	0	0	0	0	0		0	0	0	0	0	0		
0x04	RTC_DR	Reserved				YT[3:0]			YU[3:0]			WDU[2:0]			MT				
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x08	RTC_CR	Reserved				COE	OSEL [1:0]	POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRAIE	DCE	FMT	ST[2:0]	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0C	RTC_ISR	Reserved												TAMP2F	TAMP1F	TSOVF	TSF	DT [1:0]	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x10	RTC_PRER	Reserved				PREDIV_A[6:0]				Reserved	PREDIV_S[14:0]								
		1	1	1	1	1	1	1	1		0	0	0	0	0	0	0		
0x14	RTC_WUTR	Reserved												WUT[15:0]					
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
0x18	RTC_CALIBR	Reserved												DCS	DC[4:0]				
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
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]
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 78. RTC register map and reset values (continued)

Refer to [Table 1 on page 50](#) for the register boundary addresses.

23 Inter-integrated circuit (I^2C) interface

23.1 I^2C introduction

I^2C (inter-integrated circuit) bus Interface serves as an interface between the microcontroller and the serial I^2C bus. It provides multimaster capability, and controls all I^2C bus-specific sequencing, protocol, arbitration and timing. It supports standard and fast speed modes. It is also SMBus 2.0 compatible.

It may be used for a variety of purposes, including CRC generation and verification, SMBus (system management bus) and PMBus (power management bus).

Depending on specific device implementation DMA capability can be available for reduced CPU overload.

23.2 I^2C main features

- Parallel-bus/ I^2C protocol converter
- Multimaster capability: the same interface can act as Master or Slave
- I^2C Master features:
 - Clock generation
 - Start and Stop generation
- I^2C Slave features:
 - Programmable I^2C Address detection
 - Dual Addressing Capability to acknowledge 2 slave addresses
 - Stop bit detection
- Generation and detection of 7-bit/10-bit addressing and General Call
- Supports different communication speeds:
 - Standard Speed (up to 100 kHz),
 - Fast Speed (up to 400 kHz)
- Status flags:
 - Transmitter/Receiver mode flag
 - End-of-Byte transmission flag
 - I^2C busy flag
- Error flags:
 - Arbitration lost condition for master mode
 - Acknowledgement failure after address/ data transmission
 - Detection of misplaced start or stop condition
 - Overrun/Underrun if clock stretching is disabled
- 2 Interrupt vectors:
 - 1 Interrupt for successful address/ data communication
 - 1 Interrupt for error condition
- Optional clock stretching
- 1-byte buffer with DMA capability

- Configurable PEC (packet error checking) generation or verification:
 - PEC value can be transmitted as last byte in Tx mode
 - PEC error checking for last received byte
- SMBus 2.0 Compatibility:
 - 25 ms clock low timeout delay
 - 10 ms master cumulative clock low extend time
 - 25 ms slave cumulative clock low extend time
 - Hardware PEC generation/verification with ACK control
 - Address Resolution Protocol (ARP) supported
- PMBus Compatibility

Note: Some of the above features may not be available in certain products. The user should refer to the product data sheet, to identify the specific features supported by the I^2C interface implementation.

23.3 I^2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz) or fast (up to 400 kHz) I^2C bus.

23.3.1 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master, after it generates a START condition and from master to slave, if an arbitration loss or a Stop generation occurs, allowing multimaster capability.

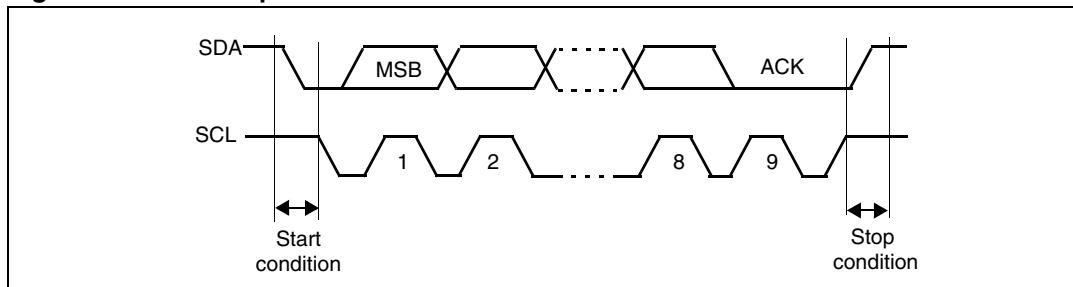
Communication flow

In Master mode, the I^2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection may be enabled or disabled by software.

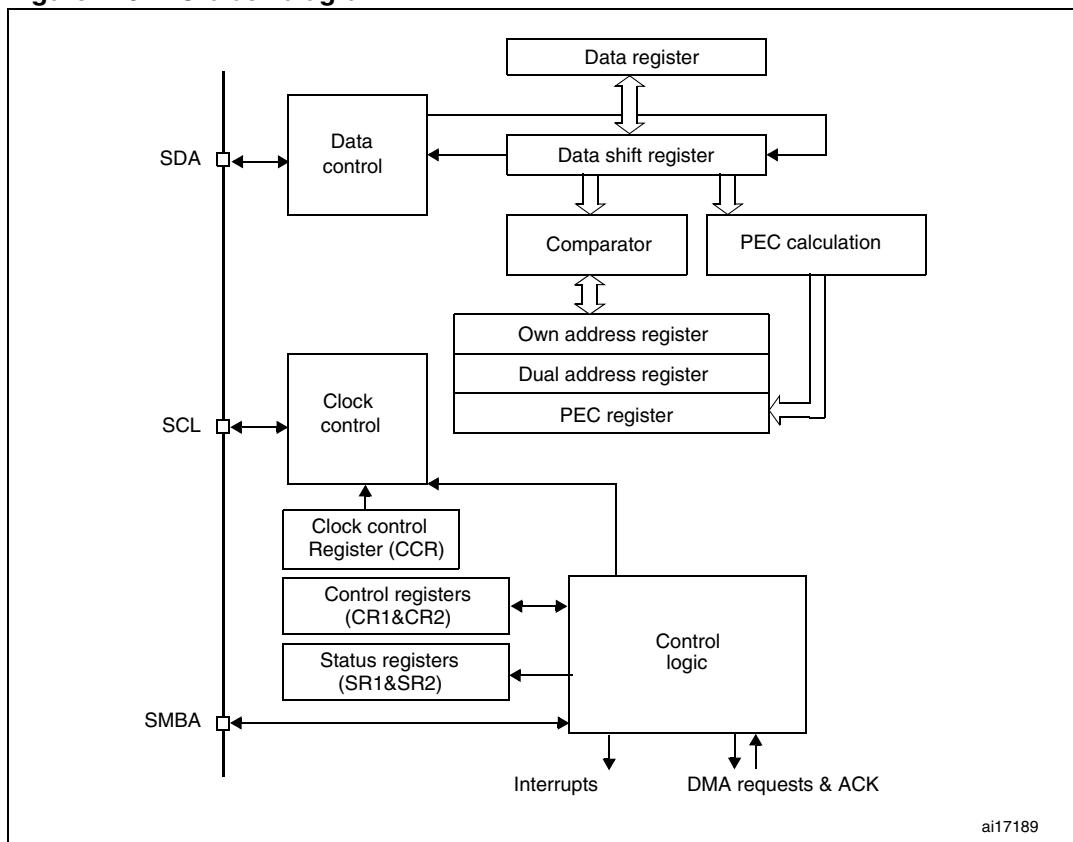
Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the start condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to [Figure 209](#).

Figure 209. I^2C bus protocol

Acknowledge may be enabled or disabled by software. The I²C interface addresses (dual addressing 7-bit/ 10-bit and/or general call address) can be selected by software.

The block diagram of the I²C interface is shown in [Figure 210](#).

Figure 210. I^2C block diagram

1. SMBA is an optional signal in SMBus mode. This signal is not applicable if SMBus is disabled.

ai17189

23.3.2 I²C slave mode

By default the I²C interface operates in Slave mode. To switch from default Slave mode to Master mode a Start condition generation is needed.

The peripheral input clock must be programmed in the I2C_CR2 register in order to generate correct timings. The peripheral input clock frequency must be at least:

- 2 MHz in Standard mode
- 4 MHz in Fast mode

As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register. Then it is compared with the address of the interface (OAR1) and with OAR2 (if ENDUAL=1) or the General Call address (if ENGC = 1).

Note: *In 10-bit addressing mode, the comparison includes the header sequence (11110xx0), where xx denotes the two most significant bits of the address.*

Header or address not matched: the interface ignores it and waits for another Start condition.

Header matched (10-bit mode only): the interface generates an acknowledge pulse if the ACK bit is set and waits for the 8-bit slave address.

Address matched: the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.
- If ENDUAL=1, the software has to read the DUALF bit to check which slave address has been acknowledged.

In 10-bit mode, after receiving the address sequence the slave is always in Receiver mode. It will enter Transmitter mode on receiving a repeated Start condition followed by the header sequence with matching address bits and the least significant bit set (11110xx1).

The TRA bit indicates whether the slave is in Receiver or Transmitter mode.

Slave transmitter

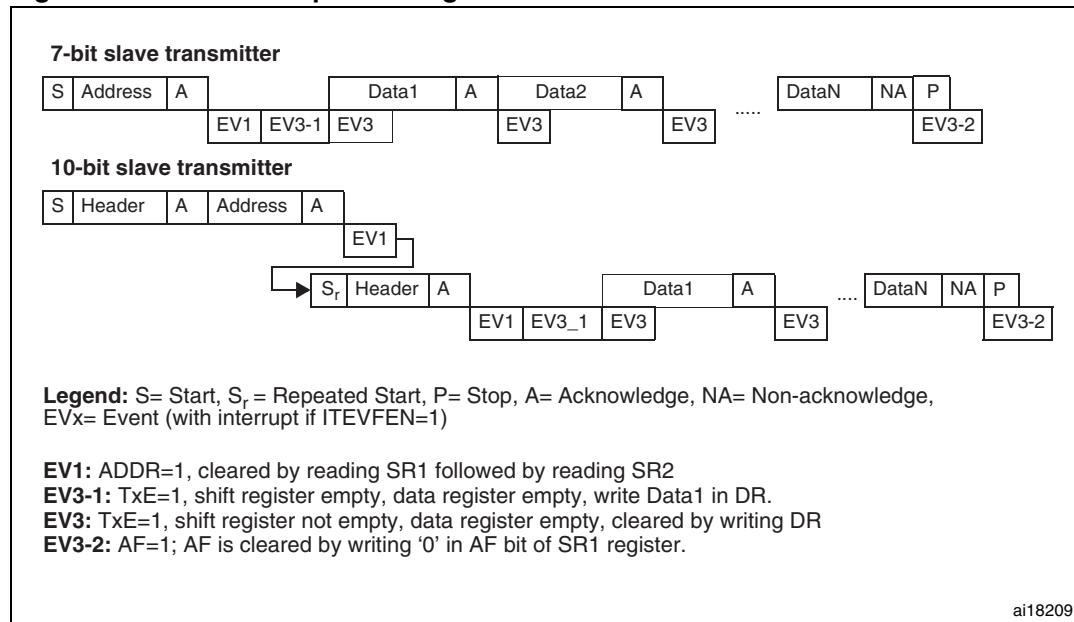
Following the address reception and after clearing ADDR, the slave sends bytes from the DR register to the SDA line via the internal shift register.

The slave stretches SCL low until ADDR is cleared and DR filled with the data to be sent (see [Figure 211 Transfer sequencing EV1 EV3](#)).

When the acknowledge pulse is received:

- The TxE bit is set by hardware with an interrupt if the ITEVFEN and the ITBUFEN bits are set.

If TxE is set and some data were not written in the I2C_DR register before the end of the next data transmission, the BTF bit is set and the interface waits until BTF is cleared by a read to I2C_SR1 followed by a write to the I2C_DR register, stretching SCL low.

Figure 211. Transfer sequence diagram for slave transmitter

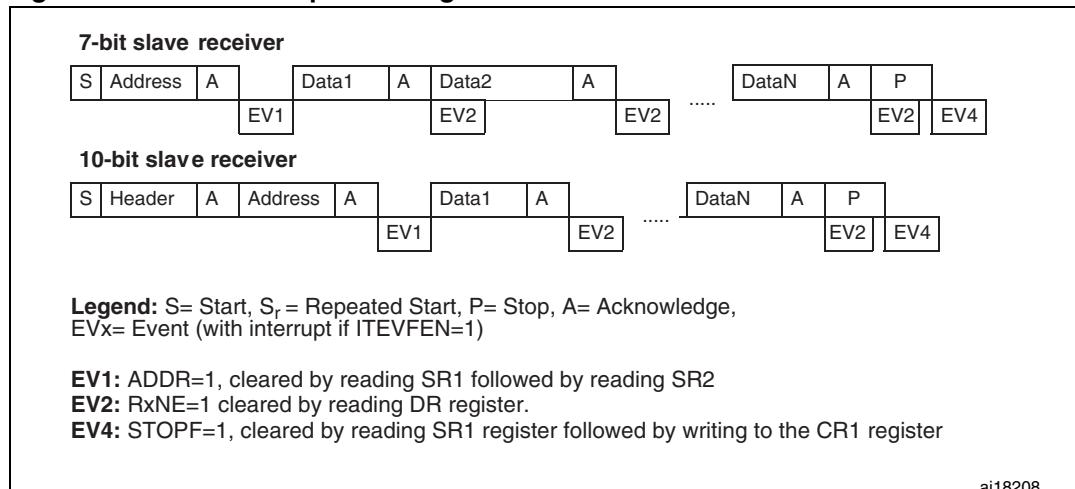
1. The EV1 and EV3_1 events stretch SCL low until the end of the corresponding software sequence.
2. The EV3 event stretches SCL low if the software sequence is not completed before the end of the next byte transmission.

Slave receiver

Following the address reception and after clearing ADDR, the slave receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The RxNE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bit is set.

If RxNE is set and the data in the DR register is not read before the end of the next data reception, the BTF bit is set and the interface waits until BTF is cleared by a read from the I2C_DR register, stretching SCL low (see [Figure 212 Transfer sequencing](#)).

Figure 212. Transfer sequence diagram for slave receiver

ai18208

1. The EV1 event stretches SCL low until the end of the corresponding software sequence.
2. The EV2 event stretches SCL low if the software sequence is not completed before the end of the next byte reception.
3. After checking the SR1 register content, the user should perform the complete clearing sequence for each flag found set.
Thus, for ADDR and STOPF flags, the following sequence is required inside the I2C interrupt routine:
READ SR1
if (ADDR == 1) {READ SR1; READ SR2}
if (STOPF == 1) {READ SR1; WRITE CR1}
The purpose is to make sure that both ADDR and STOPF flags are cleared if both are found set.

Closing slave communication

After the last data byte is transferred a Stop Condition is generated by the master. The interface detects this condition and sets:

- The STOPF bit and generates an interrupt if the ITEVFEN bit is set.

The STOPF bit is cleared by a read of the SR1 register followed by a write to the CR1 register (see [Figure 212: Transfer sequence diagram for slave receiver](#) EV4).

23.3.3 I²C master mode

In Master mode, the I²C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a Start condition and ends with a Stop condition. Master mode is selected as soon as the Start condition is generated on the bus with a START bit.

The following is the required sequence in master mode.

- Program the peripheral input clock in I2C_CR2 Register in order to generate correct timings
- Configure the clock control registers
- Configure the rise time register
- Program the I2C_CR1 register to enable the peripheral
- Set the START bit in the I2C_CR1 register to generate a Start condition

The peripheral input clock frequency must be at least:

- 2 MHz in Standard mode
- 4 MHz in Fast mode

Start condition

Setting the START bit causes the interface to generate a Start condition and to switch to Master mode (M/SL bit set) when the BUSY bit is cleared.

Note: *In master mode, setting the START bit causes the interface to generate a ReStart condition at the end of the current byte transfer.*

Once the Start condition is sent:

- The SB bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the Slave address (see [Figure 213](#) and [Figure 214](#) Transfer sequencing EV5).

Slave address transmission

Then the slave address is sent to the SDA line via the internal shift register.

- In 10-bit addressing mode, sending the header sequence causes the following event:
 - The ADD10 bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the second address byte (see [Figure 213](#) and [Figure 214](#) Transfer sequencing).

- The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see [Figure 213](#) and [Figure 214](#) Transfer sequencing).

- In 7-bit addressing mode, one address byte is sent.

As soon as the address byte is sent,

- The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see [Figure 213](#) and [Figure 214](#) Transfer sequencing).

The master can decide to enter Transmitter or Receiver mode depending on the LSB of the slave address sent.

- In 7-bit addressing mode,
 - To enter Transmitter mode, a master sends the slave address with LSB reset.
 - To enter Receiver mode, a master sends the slave address with LSB set.
- In 10-bit addressing mode,
 - To enter Transmitter mode, a master sends the header (11110xx0) and then the slave address, (where xx denotes the two most significant bits of the address).
 - To enter Receiver mode, a master sends the header (11110xx0) and then the slave address. Then it should send a repeated Start condition followed by the header (11110xx1), (where xx denotes the two most significant bits of the address).

The TRA bit indicates whether the master is in Receiver or Transmitter mode.

Master transmitter

Following the address transmission and after clearing ADDR, the master sends bytes from the DR register to the SDA line via the internal shift register.

The master waits until the first data byte is written into I2C_DR (see [Figure 213 Transfer sequencing EV8_1](#)).

When the acknowledge pulse is received, the TxE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set.

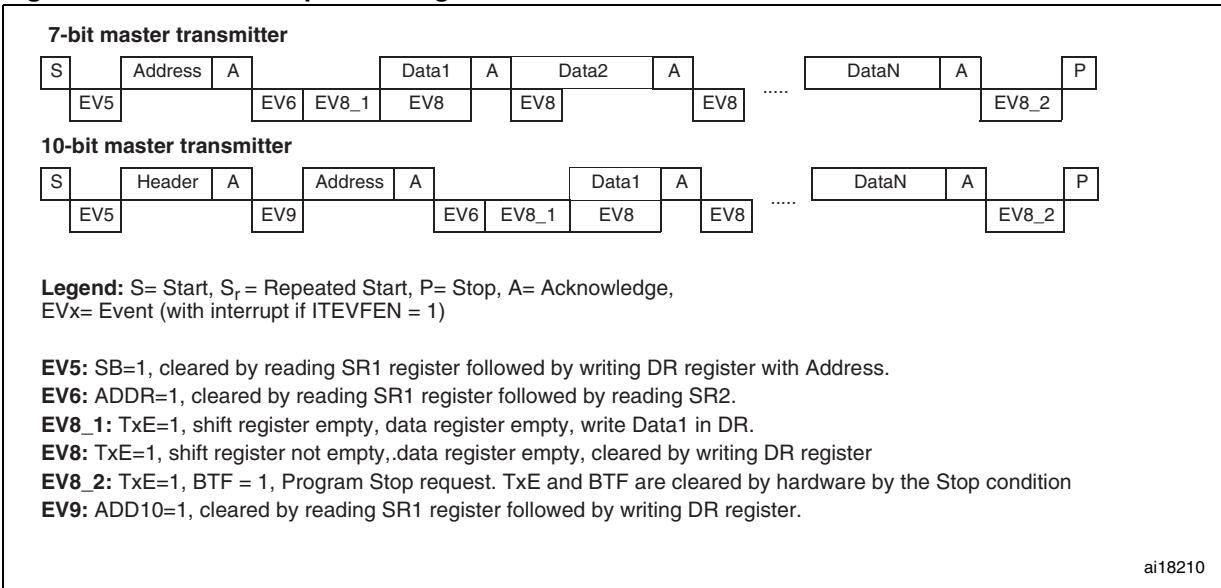
If TxE is set and a data byte was not written in the DR register before the end of the last data transmission, BTF is set and the interface waits until BTF is cleared by a write to I2C_DR, stretching SCL low.

Closing the communication

After the last byte is written to the DR register, the STOP bit is set by software to generate a Stop condition (see [Figure 213 Transfer sequencing EV8_2](#)). The interface automatically goes back to slave mode (M/SL bit cleared).

Note: Stop condition should be programmed during EV8_2 event, when either TxE or BTF is set.

Figure 213. Transfer sequence diagram for master transmitter



1. The EV5, EV6, EV9, EV8_1 and EV8_2 events stretch SCL low until the end of the corresponding software sequence.
2. The EV8 event stretches SCL low if the software sequence is not complete before the end of the next byte transmission.

Master receiver

Following the address transmission and after clearing ADDR, the I^2C interface enters Master Receiver mode. In this mode the interface receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

1. An acknowledge pulse if the ACK bit is set
2. The RxNE bit is set and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set (see [Figure 214 Transfer sequencing EV7](#)).

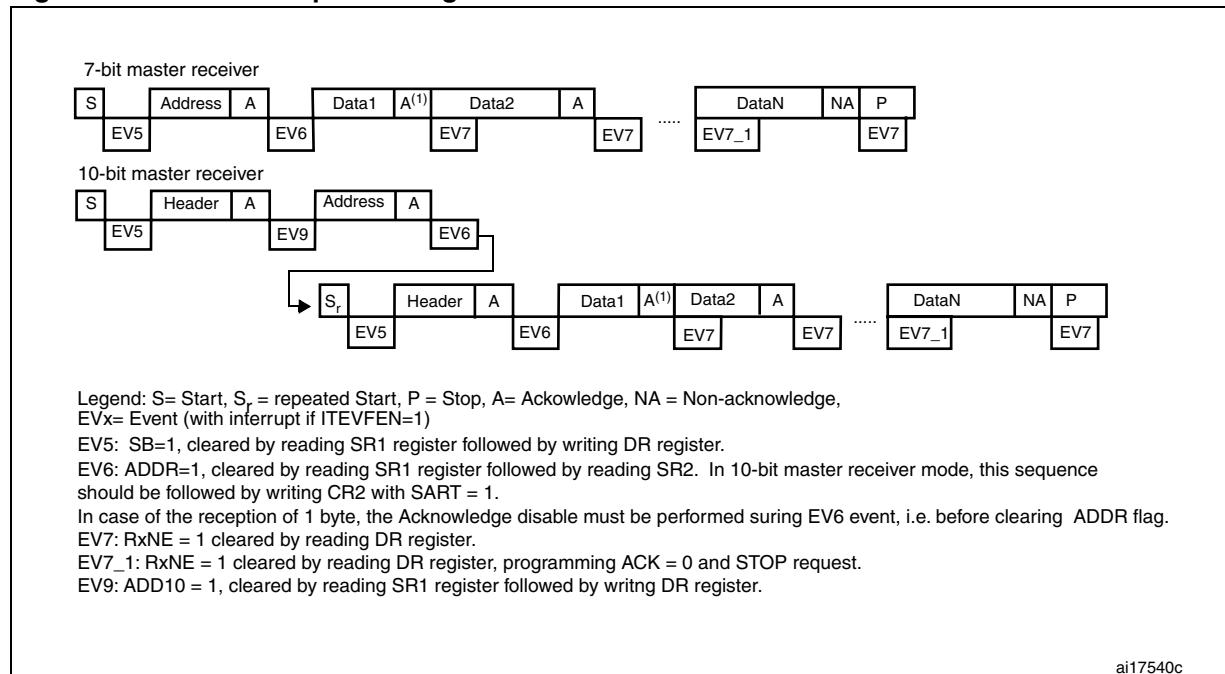
If the RxNE bit is set and the data in the DR register is not read before the end of the last data reception, the BTF bit is set by hardware and the interface waits until BTF is cleared by a read in the DR register, stretching SCL low.

Closing the communication

The master sends a NACK for the last byte received from the slave. After receiving this NACK, the slave releases the control of the SCL and SDA lines. Then the master can send a Stop/Restart condition.

1. To generate the nonacknowledge pulse after the last received data byte, the ACK bit must be cleared just after reading the second last data byte (after second last RxNE event).
2. In order to generate the Stop/Restart condition, software must set the STOP/START bit after reading the second last data byte (after the second last RxNE event).
3. In case a single byte has to be received, the Acknowledge disable is made during EV6 (before ADDR flag is cleared) and the STOP condition generation is made after EV6.

After the Stop condition generation, the interface goes automatically back to slave mode (M/SL bit cleared).

Figure 214. Transfer sequence diagram for master receiver

ai17540c

1. If a single byte is received, it is NA.
2. The EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.
3. The EV7 event stretches SCL low if the software sequence is not completed before the end of the next byte reception.
4. The EV7_1 software sequence must be completed before the ACK pulse of the current byte transfer.

The procedures described below are recommended if the EV7-1 software sequence is not completed before the ACK pulse of the current byte transfer.

These procedures must be followed to make sure:

- The ACK bit is set low on time before the end of the last data reception
- The STOP bit is set high after the last data reception without reception of supplementary data.

For 2-byte reception:

- Wait until ADDR = 1 (SCL stretched low until the ADDR flag is cleared)
- Set ACK low, set POS high
- Clear ADDR flag
- Wait until BTF = 1 (Data 1 in DR, Data2 in shift register, SCL stretched low until a data 1 is read)
- Set STOP high
- Read data 1 and 2

For N > 2 -byte reception, from N-2 data reception

- Wait until BTF = 1 (data N-2 in DR, data N-1 in shift register, SCL stretched low until data N-2 is read)
- Set ACK low
- Read data N-2
- Wait until BTF = 1 (data N-1 in DR, data N in shift register, SCL stretched low until a data N-1 is read)
- Set STOP high
- Read data N-1 and N

23.3.4 Error conditions

The following are the error conditions which may cause communication to fail.

Bus error (BERR)

This error occurs when the I^2C interface detects an external Stop or Start condition during an address or a data transfer. In this case:

- the BERR bit is set and an interrupt is generated if the ITERREN bit is set
- in Slave mode: data are discarded and the lines are released by hardware:
 - in case of a misplaced Start, the slave considers it is a restart and waits for an address, or a Stop condition
 - in case of a misplaced Stop, the slave behaves like for a Stop condition and the lines are released by hardware
- In Master mode: the lines are not released and the state of the current transmission is not affected. It is up to the software to abort or not the current transmission

Acknowledge failure (AF)

This error occurs when the interface detects a nonacknowledge bit. In this case:

- the AF bit is set and an interrupt is generated if the ITERREN bit is set
- a transmitter which receives a NACK must reset the communication:
 - If Slave: lines are released by hardware
 - If Master: a Stop or repeated Start condition must be generated by software

Arbitration lost (ARLO)

This error occurs when the I^2C interface detects an arbitration lost condition. In this case,

- the ARLO bit is set by hardware (and an interrupt is generated if the ITERREN bit is set)
- the I^2C Interface goes automatically back to slave mode (the M/SL bit is cleared). When the I^2C loses the arbitration, it is not able to acknowledge its slave address in the same transfer, but it can acknowledge it after a repeated Start from the winning master.
- lines are released by hardware

Overrun/underrun error (OVR)

An overrun error can occur in slave mode when clock stretching is disabled and the I²C interface is receiving data. The interface has received a byte (RxNE=1) and the data in DR has not been read, before the next byte is received by the interface. In this case,

- The last received byte is lost.
- In case of Overrun error, software should clear the RxNE bit and the transmitter should re-transmit the last received byte.

Underrun error can occur in slave mode when clock stretching is disabled and the I²C interface is transmitting data. The interface has not updated the DR with the next byte (TxE=1), before the clock comes for the next byte. In this case,

- The same byte in the DR register will be sent again
- The user should make sure that data received on the receiver side during an underrun error are discarded and that the next bytes are written within the clock low time specified in the I²C bus standard.

For the first byte to be transmitted, the DR must be written after ADDR is cleared and before the first SCL rising edge. If not possible, the receiver must discard the first data.

23.3.5 SDA/SCL line control

- If clock stretching is enabled:
 - Transmitter mode: If TxE=1 and BTF=1: the interface holds the clock line low before transmission to wait for the microcontroller to write the byte in the Data Register (both buffer and shift register are empty).
 - Receiver mode: If RxNE=1 and BTF=1: the interface holds the clock line low after reception to wait for the microcontroller to read the byte in the Data Register (both buffer and shift register are full).
- If clock stretching is disabled in Slave mode:
 - Overrun Error in case of RxNE=1 and no read of DR has been done before the next byte is received. The last received byte is lost.
 - Underrun Error in case TxE=1 and no write into DR has been done before the next byte must be transmitted. The same byte will be sent again.
 - Write Collision not managed.

23.3.6 SMBus

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I²C principles of operation. SMBus provides a control bus for system and power management related tasks. A system may use SMBus to pass messages to and from devices instead of toggling individual control lines.

The System Management Bus Specification refers to three types of devices. A *slave* is a device that is receiving or responding to a command. A *master* is a device that issues commands, generates the clocks, and terminates the transfer. A *host* is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

Similarities between SMBus and I²C

- 2 wire bus protocol (1 Clk, 1 Data) + SMBus Alert line optional
- Master-slave communication, Master provides clock
- Multi master capability
- SMBus data format similar to I²C 7-bit addressing format ([Figure 209](#)).

Differences between SMBus and I²C

The following table describes the differences between SMBus and I²C.

Table 79. SMBus vs. I²C

SMBus	I ² C
Max. speed 100 kHz	Max. speed 400 kHz
Min. clock speed 10 kHz	No minimum clock speed
35 ms clock low timeout	No timeout
Logic levels are fixed	Logic levels are V _{DD} dependent
Different address types (reserved, dynamic etc.)	7-bit, 10-bit and general call slave address types
Different bus protocols (quick command, process call etc.)	No bus protocols

SMBus application usage

With System Management Bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status. SMBus provides a control bus for system and power management related tasks.

Device identification

Any device that exists on the System Management Bus as a slave has a unique address called the Slave Address. For the list of reserved slave addresses, refer to the SMBus specification ver. 2.0 (<http://smbus.org/specs/>).

Bus protocols

The SMBus specification supports up to 9 bus protocols. For more details of these protocols and SMBus address types, refer to SMBus specification ver. 2.0 (<http://smbus.org/specs/>). These protocols should be implemented by the user software.

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. The Address Resolution Protocol (ARP) has the following attributes:

- Address assignment uses the standard SMBus physical layer arbitration mechanism
- Assigned addresses remain constant while device power is applied; address retention through device power loss is also allowed
- No additional SMBus packet overhead is incurred after address assignment. (i.e. subsequent accesses to assigned slave addresses have the same overhead as accesses to fixed address devices.)
- Any SMBus master can enumerate the bus

Unique device identifier (UDID)

In order to provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique device identifier (UDID).

For the details on 128 bit UDID and more information on ARP, refer to SMBus specification ver. 2.0 (<http://smbus.org/specs>).

SMBus alert mode

SMBus Alert is an optional signal with an interrupt line for devices that want to trade their ability to master for a pin. SMBA is a wired-AND signal just as the SCL and SDA signals are. SMBA is used in conjunction with the SMBus General Call Address. Messages invoked with the SMBus are 2 bytes long.

A slave-only device can signal the host through SMBA that it wants to talk by setting ALERT bit in I2C_CR1 register. The host processes the interrupt and simultaneously accesses all SMBA devices through the *Alert Response Address* (known as ARA having a value 0001 100X). Only the device(s) which pulled SMBA low will acknowledge the Alert Response Address. This status is identified using SMBALERT Status flag in I2C_SR1 register. The host performs a modified Receive Byte operation. The 7 bit device address provided by the slave transmit device is placed in the 7 most significant bits of the byte. The eighth bit can be a zero or one.

If more than one device pulls SMBA low, the highest priority (lowest address) device will win communication rights via standard arbitration during the slave address transfer. After acknowledging the slave address the device must disengage its SMBA pull-down. If the host still sees SMBA low when the message transfer is complete, it knows to read the ARA again.

A host which does not implement the SMBA signal may periodically access the ARA.

For more details on SMBus Alert mode, refer to SMBus specification ver. 2.0 (<http://smbus.org/specs>).

Timeout error

There are differences in the timing specifications between I²C and SMBus.

SMBus defines a clock low timeout, TIMEOUT of 35 ms. Also SMBus specifies TLOW: SEXT as the cumulative clock low extend time for a slave device. SMBus specifies TLOW: MEXT as the cumulative clock low extend time for a master device. For more details on these timeouts, refer to SMBus specification ver. 2.0 (<http://smbus.org/specs>).

The status flag Timeout or Tlow Error in I2C_SR1 shows the status of this feature.

How to use the interface in SMBus mode

To switch from I²C mode to SMBus mode, the following sequence should be performed.

- Set the SMBus bit in the I2C_CR1 register
- Configure the SMBTYPE and ENARP bits in the I2C_CR1 register as required for the application

If you want to configure the device as a master, follow the Start condition generation procedure in [Section 23.3.3: I²C master mode](#). Otherwise, follow the sequence in [Section 23.3.2: I²C slave mode](#).

The application has to control the various SMBus protocols by software.

- SMB Device Default Address acknowledged if ENARP=1 and SMBTYPE=0
- SMB Host Header acknowledged if ENARP=1 and SMBTYPE=1
- SMB Alert Response Address acknowledged if SMBALERT=1

23.3.7 DMA requests

DMA requests (when enabled) are generated only for data transfer. DMA requests are generated by Data Register becoming empty in transmission and Data Register becoming full in reception. The DMA request must be served before the end of the current byte transfer. When the number of data transfers which has been programmed for the corresponding DMA channel is reached, the DMA controller sends an End of Transfer EOT signal to the I²C interface and generates a Transfer Complete interrupt if enabled:

- Master transmitter: In the interrupt routine after the EOT interrupt, disable DMA requests then wait for a BTF event before programming the Stop condition.
- Master receiver
 - When the number of bytes to be received is equal to or greater than two, the DMA controller sends a hardware signal, EOT_1, corresponding to the last but one data byte (number_of_bytes – 1). If, in the I2C_CR2 register, the LAST bit is set, I²C automatically sends a NACK after the next byte following EOT_1. The user can generate a Stop condition in the DMA Transfer Complete interrupt routine if enabled.
 - When a single byte must be received: the NACK must be programmed during EV6 event, i.e. program ACK=0 when ADDR=1, before clearing ADDR flag. Then the user can program the STOP condition either after clearing ADDR flag, or in the DMA Transfer Complete interrupt routine.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAEN bit in the I2C_CR2 register. Data will be loaded from a Memory area configured using the DMA peripheral (refer to the DMA specification) to the I2C_DR register whenever the TxE bit is set. To map a DMA channel for I²C transmission, perform the following sequence. Here x is the channel number.

1. Set the I2C_DR register address in the DMA_CPARx register. The data will be moved to this address from the memory after each TxE event.
2. Set the memory address in the DMA_CMARx register. The data will be loaded into I2C_DR from this memory after each TxE event.
3. Configure the total number of bytes to be transferred in the DMA_CNDTRx register. After each TxE event, this value will be decremented.
4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register
5. Set the DIR bit and, in the DMA_CCRx register, configure interrupts after half transfer or full transfer depending on application requirements.
6. Activate the channel by setting the EN bit in the DMA_CCRx register.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I²C interface and the DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.

Note: *Do not enable the ITBUFEN bit in the I2C_CR2 register if DMA is used for transmission.*

Reception using DMA

DMA mode can be enabled for reception by setting the DMAEN bit in the I2C_CR2 register. Data will be loaded from the I2C_DR register to a Memory area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA channel for I²C reception, perform the following sequence. Here x is the channel number.

1. Set the I2C_DR register address in DMA_CPARx register. The data will be moved from this address to the memory after each RxNE event.
2. Set the memory address in the DMA_CMARx register. The data will be loaded from the I2C_DR register to this memory area after each RxNE event.
3. Configure the total number of bytes to be transferred in the DMA_CNDTRx register. After each RxNE event, this value will be decremented.
4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register
5. Reset the DIR bit and configure interrupts in the DMA_CCRx register after half transfer or full transfer depending on application requirements.
6. Activate the channel by setting the EN bit in the DMA_CCRx register.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I²C interface and DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.

Note: *Do not enable the ITBUFEN bit in the I2C_CR2 register if DMA is used for reception.*

23.3.8 Packet error checking

A PEC calculator has been implemented to improve the reliability of communication. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial serially on each bit.

- PEC calculation is enabled by setting the ENPEC bit in the I2C_CR1 register. PEC is a CRC-8 calculated on all message bytes including addresses and R/W bits.
 - In transmission: set the PEC transfer bit in the I2C_CR1 register after the TxE event corresponding to the last byte. The PEC will be transferred after the last transmitted byte.
 - In reception: set the PEC bit in the I2C_CR1 register after the RxNE event corresponding to the last byte so that the receiver sends a NACK if the next received byte is not equal to the internally calculated PEC. In case of Master-Receiver, a NACK must follow the PEC whatever the check result. The PEC must be set before the ACK of the CRC reception in slave mode. It must be set when the ACK is set low in master mode.
- A PECERR error flag/interrupt is also available in the I2C_SR1 register.
- If DMA and PEC calculation are both enabled:
 - In transmission: when the I²C interface receives an EOT signal from the DMA controller, it automatically sends a PEC after the last byte.
 - In reception: when the I²C interface receives an EOT_1 signal from the DMA controller, it will automatically consider the next byte as a PEC and will check it. A DMA request is generated after PEC reception.
- To allow intermediate PEC transfers, a control bit is available in the I2C_CR2 register (LAST bit) to determine if it is really the last DMA transfer or not. If it is the last DMA request for a master receiver, a NACK is automatically sent after the last received byte.
- PEC calculation is corrupted by an arbitration loss.

23.4 I²C interrupts

The table below gives the list of I²C interrupt requests.

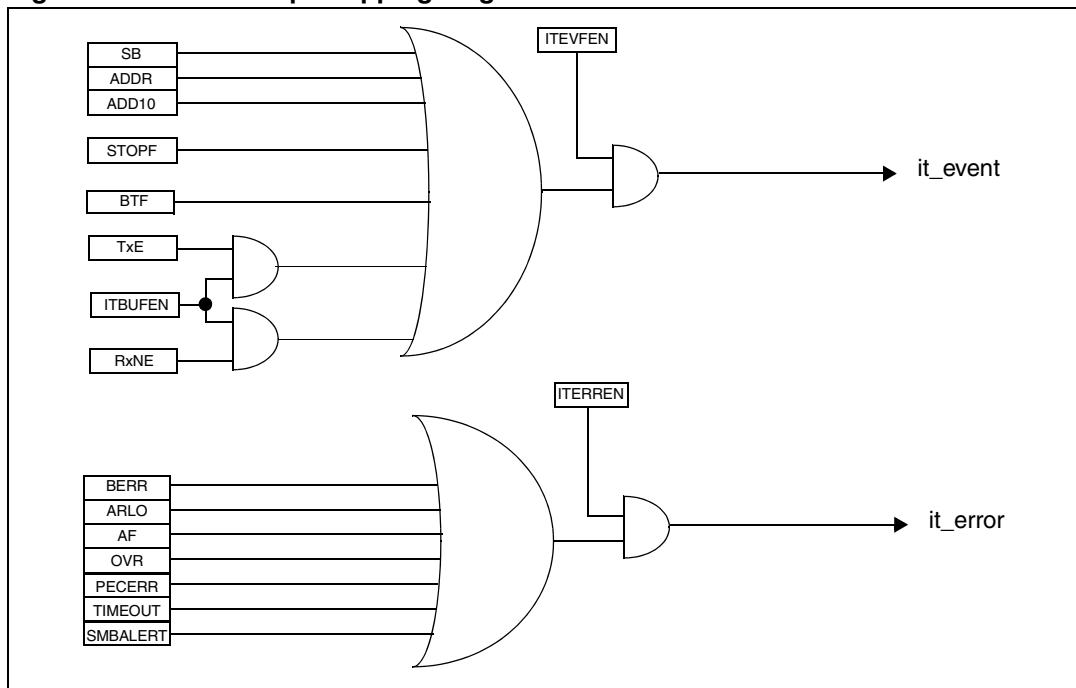
Table 80. I²C Interrupt requests

Interrupt event	Event flag	Enable control bit
Start bit sent (Master)	SB	ITEVFEN
Address sent (Master) or Address matched (Slave)	ADDR	
10-bit header sent (Master)	ADD10	
Stop received (Slave)	STOPF	
Data byte transfer finished	BTF	
Receive buffer not empty	RxNE	ITEVFEN and ITBUFEN
Transmit buffer empty	TxE	

Table 80. I²C Interrupt requests (continued)

Interrupt event	Event flag	Enable control bit
Bus error	BERR	ITERREN
Arbitration loss (Master)	ARLO	
Acknowledge failure	AF	
Overrun/Underrun	OVR	
PEC error	PECERR	
Timeout/Tlow error	TIMEOUT	
SMBus Alert	SMBALERT	

- Note:
- 1 *SB, ADDR, ADD10, STOPF, BTF, RxNE and TxE are logically ORed on the same interrupt channel.*
 - 2 *BERR, ARLO, AF, OVR, PECERR, TIMEOUT and SMBALERT are logically ORed on the same interrupt channel.*

Figure 215. I²C interrupt mapping diagram

23.5 I²C debug mode

When the microcontroller enters the debug mode (Cortex™-M4F core halted), the SMBUS timeout either continues to work normally or stops, depending on the DBG_I2Cx_SMBUS_TIMEOUT configuration bits in the DBG module. For more details, refer to [Section 32.16.2: Debug support for timers, watchdog, bxCAN and I²C on page 1297](#).

23.6 I²C registers

Refer to [Section 1.1 on page 46](#) for a list of abbreviations used in register descriptions.

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

23.6.1 I²C Control register 1 (I2C_CR1)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res.	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENG C	ENPEC	ENARP	SMB TYPE	Res.	SMBUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 15 **SWRST**: Software reset

When set, the I²C is under reset state. Before resetting this bit, make sure the I²C lines are released and the bus is free.

- 0: I²C Peripheral not under reset
- 1: I²C Peripheral under reset state

Note: This bit can be used in case the BUSY bit is set to '1' when no stop condition has been detected on the bus.

Bit 14 Reserved, must be kept at reset value

Bit 13 **ALERT**: SMBus alert

This bit is set and cleared by software, and cleared by hardware when PE=0.

- 0: Releases SMBA pin high. Alert Response Address Header followed by NACK.
- 1: Drives SMBA pin low. Alert Response Address Header followed by ACK.

Bit 12 **PEC**: Packet error checking

This bit is set and cleared by software, and cleared by hardware when PEC is transferred or by a START or Stop condition or when PE=0.

- 0: No PEC transfer
- 1: PEC transfer (in Tx or Rx mode)

Note: PEC calculation is corrupted by an arbitration loss.

Bit 11 POS: Acknowledge/PEC Position (for data reception)

This bit is set and cleared by software and cleared by hardware when PE=0.

0: ACK bit controls the (N)ACK of the current byte being received in the shift register. The PEC bit indicates that current byte in shift register is a PEC.

1: ACK bit controls the (N)ACK of the next byte which will be received in the shift register. The PEC bit indicates that the next byte in the shift register is a PEC

Note: The POS bit must be used only in 2-byte reception configuration in master mode. It must be configured before data reception starts, as described in the 2-byte reception procedure recommended in [Section : Master receiver on page 583](#).

Note:

Bit 10 ACK: Acknowledge enable

This bit is set and cleared by software and cleared by hardware when PE=0.

0: No acknowledge returned

1: Acknowledge returned after a byte is received (matched address or data)

Bit 9 STOP: Stop generation

The bit is set and cleared by software, cleared by hardware when a Stop condition is detected, set by hardware when a timeout error is detected.

In Master Mode:

0: No Stop generation.

1: Stop generation after the current byte transfer or after the current Start condition is sent.

In Slave mode:

0: No Stop generation.

1: Release the SCL and SDA lines after the current byte transfer.

Note: When the STOP, START or PEC bit is set, the software must not perform any write access to I2C_CR1 before this bit is cleared by hardware. Otherwise there is a risk of setting a second STOP, START or PEC request.

Bit 8 START: Start generation

This bit is set and cleared by software and cleared by hardware when start is sent or PE=0.

In Master Mode:

0: No Start generation

1: Repeated start generation

In Slave mode:

0: No Start generation

1: Start generation when the bus is free

Bit 7 NOSTRETCH: Clock stretching disable (Slave mode)

This bit is used to disable clock stretching in slave mode when ADDR or BTF flag is set, until it is reset by software.

0: Clock stretching enabled

1: Clock stretching disabled

Bit 6 ENGC: General call enable

0: General call disabled. Address 00h is NACKed.

1: General call enabled. Address 00h is ACKed.

Bit 5 ENPEC: PEC enable

0: PEC calculation disabled

1: PEC calculation enabled

- Bit 4 **ENARP**: ARP enable
 0: ARP disable
 1: ARP enable
 SMBus Device default address recognized if SMBTYPE=0
 SMBus Host address recognized if SMBTYPE=1
- Bit 3 **SMBTYPE**: SMBus type
 0: SMBus Device
 1: SMBus Host
- Bit 2 Reserved, must be kept at reset value
- Bit 1 **SMBUS**: SMBus mode
 0: I²C mode
 1: SMBus mode
- Bit 0 **PE**: Peripheral enable
 0: Peripheral disable
 1: Peripheral enable: the corresponding IOs are selected as alternate functions depending on SMBus bit.
- Note: If this bit is reset while a communication is on going, the peripheral is disabled at the end of the current communication, when back to IDLE state.
 All bit resets due to PE=0 occur at the end of the communication.
 In master mode, this bit must not be reset before the end of the communication.*

23.6.2 I²C Control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0									
Reserved	<table border="1"> <tr> <td>LAST</td><td>DMA EN</td><td>ITBUF EN</td><td>ITEVT EN</td><td>ITERR EN</td> </tr> <tr> <td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td> </tr> </table>					LAST	DMA EN	ITBUF EN	ITEVT EN	ITERR EN	rw	rw	rw	rw	rw	Reserved	FREQ[5:0]								
LAST	DMA EN	ITBUF EN	ITEVT EN	ITERR EN																					
rw	rw	rw	rw	rw																					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw											

Bits 15:13 Reserved, must be kept at reset value

Bit 12 **LAST**: DMA last transfer

- 0: Next DMA EOT is not the last transfer
 1: Next DMA EOT is the last transfer

Note: This bit is used in master receiver mode to permit the generation of a NACK on the last received data.

Bit 11 **DMAEN**: DMA requests enable

- 0: DMA requests disabled
 1: DMA request enabled when TxE=1 or RxNE =1

Bit 10 **ITBUFEN**: Buffer interrupt enable

- 0: TxE = 1 or RxNE = 1 does not generate any interrupt.
 1:TxE = 1 or RxNE = 1 generates Event Interrupt (whatever the state of DMAEN)

Bit 9 **ITEVTEN**: Event interrupt enable

- 0: Event interrupt disabled
- 1: Event interrupt enabled

This interrupt is generated when:

- SB = 1 (Master)
- ADDR = 1 (Master/Slave)
- ADD10= 1 (Master)
- STOPF = 1 (Slave)
- BTF = 1 with no TxE or RxNE event
- TxE event to 1 if ITBUFEN = 1
- RxNE event to 1if ITBUFEN = 1

Bit 8 **ITERREN**: Error interrupt enable

- 0: Error interrupt disabled
- 1: Error interrupt enabled

This interrupt is generated when:

- BERR = 1
- ARLO = 1
- AF = 1
- OVR = 1
- PECERR = 1
- TIMEOUT = 1
- SMBALERT = 1

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

Bits 5:0 **FREQ[5:0]**: Peripheral clock frequency

The peripheral clock frequency must be configured using the input APB clock frequency (I²C peripheral connected to APB). The minimum allowed frequency is 2 MHz, the maximum frequency is limited by the maximum APB frequency 42 MHzand an intrinsic limitation of 46 MHz.

0b000000: Not allowed

0b000001: Not allowed

0b000010: 2 MHz

...

0b101010: 42MHz

Higher than 0b101010: Not allowed

23.6.3 I²C Own address register 1 (I²C_OAR1)

Address offset: 0x08

Reset value: 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ADD MODE	Reserved							ADD[9:8]		ADD[7:1]							ADD0
rw								rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit 15 **ADDMODE**: Addressing mode (slave mode)

0: 7-bit slave address (10-bit address not acknowledged)

1: 10-bit slave address (7-bit address not acknowledged)

Bit 14 Reserved, must be kept at reset value

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

Bits 9:8 **ADD[9:8]**: Interface address

7-bit addressing mode: don't care

10-bit addressing mode: bits 9:8 of address

Bits 7:1 **ADD[7:1]**: Interface address

bits 7:1 of address

Bit 0 **ADD0**: Interface address

7-bit addressing mode: don't care

10-bit addressing mode: bit 0 of address

23.6.4 I²C Own address register 2 (I²C_OAR2)

Address offset: 0x0C

Reset value: 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved							ADD2[7:1]							ENDUAL	
								rw	rw	rw	rw	rw	rw	rw	rw	

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

Bits 7:1 **ADD2[7:1]**: Interface address

bits 7:1 of address in dual addressing mode

Bit 0 **ENDUAL**: Dual addressing mode enable

0: Only OAR1 is recognized in 7-bit addressing mode

1: Both OAR1 and OAR2 are recognized in 7-bit addressing mode

23.6.5 I²C Data register (I²C_DR)

Address offset: 0x10

Reset value: 0x0000

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

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

Bits 7:0 DR[7:0] 8-bit data register

Byte received or to be transmitted to the bus.

– Transmitter mode: Byte transmission starts automatically when a byte is written in the DR register. A continuous transmit stream can be maintained if the next data to be transmitted is put in DR once the transmission is started (TxE=1)

– Receiver mode: Received byte is copied into DR (RxNE=1). A continuous transmit stream can be maintained if DR is read before the next data byte is received (RxNE=1).

Note: In slave mode, the address is not copied into DR.

Note: Write collision is not managed (DR can be written if TxE=0).

Note: If an ARLO event occurs on ACK pulse, the received byte is not copied into DR and so cannot be read.

23.6.6 I²C Status register 1 (I²C_SR1)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMB ALERT	TIME OUT	Res.	PEC ERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOPF	ADD10	BTF	ADDR	SB
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

Bit 15 SMBALERT: SMBus alert

In SMBus host mode:

0: no SMBALERT

1: SMBALERT event occurred on pin

In SMBus slave mode:

0: no SMBALERT response address header

1: SMBALERT response address header to SMBALERT LOW received

– Cleared by software writing 0, or by hardware when PE=0.

Bit 14 TIMEOUT: Timeout or Tlow error

0: No timeout error

1: SCL remained LOW for 25 ms (Timeout)

or

Master cumulative clock low extend time more than 10 ms (Tlow:mext)

or

Slave cumulative clock low extend time more than 25 ms (Tlow:sext)

– When set in slave mode: slave resets the communication and lines are released by hardware

– When set in master mode: Stop condition sent by hardware

– Cleared by software writing 0, or by hardware when PE=0.

Note: This functionality is available only in SMBus mode.

Bit 13 Reserved, must be kept at reset value

Bit 12 **PECERR**: PEC Error in reception

0: no PEC error: receiver returns ACK after PEC reception (if ACK=1)

1: PEC error: receiver returns NACK after PEC reception (whatever ACK)

–Cleared by software writing 0, or by hardware when PE=0.

–Note: When the received CRC is wrong, PECERR is not set in slave mode if the PEC control bit is not set before the end of the CRC reception. Nevertheless, reading the PEC value determines whether the received CRC is right or wrong.

Bit 11 **OVR**: Overrun/Underrun

0: No overrun/underrun

1: Overrun or underrun

–Set by hardware in slave mode when NOSTRETCH=1 and:

–In reception when a new byte is received (including ACK pulse) and the DR register has not been read yet. New received byte is lost.

–In transmission when a new byte should be sent and the DR register has not been written yet. The same byte is sent twice.

–Cleared by software writing 0, or by hardware when PE=0.

Note: If the DR write occurs very close to SCL rising edge, the sent data is unspecified and a hold timing error occurs

Bit 10 **AF**: Acknowledge failure

0: No acknowledge failure

1: Acknowledge failure

–Set by hardware when no acknowledge is returned.

–Cleared by software writing 0, or by hardware when PE=0.

Bit 9 **ARLO**: Arbitration lost (master mode)

0: No Arbitration Lost detected

1: Arbitration Lost detected

Set by hardware when the interface loses the arbitration of the bus to another master

–Cleared by software writing 0, or by hardware when PE=0.

After an ARLO event the interface switches back automatically to Slave mode (M/SL=0).

Note: In SMBUS, the arbitration on the data in slave mode occurs only during the data phase, or the acknowledge transmission (not on the address acknowledge).

Bit 8 **BERR**: Bus error

0: No misplaced Start or Stop condition

1: Misplaced Start or Stop condition

–Set by hardware when the interface detects an SDA rising or falling edge while SCL is high, occurring in a non-valid position during a byte transfer.

–Cleared by software writing 0, or by hardware when PE=0.

Bit 7 **TxE**: Data register empty (transmitters)

0: Data register not empty

1: Data register empty

–Set when DR is empty in transmission. TxE is not set during address phase.

–Cleared by software writing to the DR register or by hardware after a start or a stop condition or when PE=0.

TxE is not set if either a NACK is received, or if next byte to be transmitted is PEC (PEC=1)

Note: TxE is not cleared by writing the first data being transmitted, or by writing data when BTF is set, as in both cases the data register is still empty.

Bit 6 **RxNE**: Data register not empty (receivers)

- 0: Data register empty
- 1: Data register not empty

–Set when data register is not empty in receiver mode. RxNE is not set during address phase.

–Cleared by software reading or writing the DR register or by hardware when PE=0.

RxNE is not set in case of ARLO event.

Note: RxNE is not cleared by reading data when BTF is set, as the data register is still full.

Bit 5 Reserved, must be kept at reset value

Bit 4 **STOPF**: Stop detection (slave mode)

- 0: No Stop condition detected
- 1: Stop condition detected

–Set by hardware when a Stop condition is detected on the bus by the slave after an acknowledge (if ACK=1).

–Cleared by software reading the SR1 register followed by a write in the CR1 register, or by hardware when PE=0

Note: The STOPF bit is not set after a NACK reception.

It is recommended to perform the complete clearing sequence (READ SR1 then WRITE CR1) after the STOPF is set. Refer to [Figure 212: Transfer sequence diagram for slave receiver on page 580](#).

Bit 3 **ADD10**: 10-bit header sent (Master mode)

- 0: No ADD10 event occurred.
- 1: Master has sent first address byte (header).

–Set by hardware when the master has sent the first byte in 10-bit address mode.

–Cleared by software reading the SR1 register followed by a write in the DR register of the second address byte, or by hardware when PE=0.

Note: ADD10 bit is not set after a NACK reception

Bit 2 **BTF**: Byte transfer finished

- 0: Data byte transfer not done
- 1: Data byte transfer succeeded

–Set by hardware when NOSTRETCH=0 and:

–In reception when a new byte is received (including ACK pulse) and DR has not been read yet (RxNE=1).

–In transmission when a new byte should be sent and DR has not been written yet (TxE=1).

–Cleared by software by either a read or write in the DR register or by hardware after a start or a stop condition in transmission or when PE=0.

Note: The BTF bit is not set after a NACK reception

The BTF bit is not set if next byte to be transmitted is the PEC (TRA=1 in I2C_SR2 register and PEC=1 in I2C_CR1 register)

Bit 1 **ADDR**: Address sent (master mode)/matched (slave mode)

This bit is cleared by software reading SR1 register followed reading SR2, or by hardware when PE=0.

Address matched (Slave)

0: Address mismatched or not received.

1: Received address matched.

–Set by hardware as soon as the received slave address matched with the OAR registers content or a general call or a SMBus Device Default Address or SMBus Host or SMBus Alert is recognized. (when enabled depending on configuration).

Note: In slave mode, it is recommended to perform the complete clearing sequence (READ SR1 then READ SR2) after ADDR is set. Refer to [Figure 212: Transfer sequence diagram for slave receiver on page 580](#).

Address sent (Master)

0: No end of address transmission

1: End of address transmission

–For 10-bit addressing, the bit is set after the ACK of the 2nd byte.

–For 7-bit addressing, the bit is set after the ACK of the byte.

Note: ADDR is not set after a NACK reception

Bit 0 **SB**: Start bit (Master mode)

0: No Start condition

1: Start condition generated.

–Set when a Start condition generated.

–Cleared by software by reading the SR1 register followed by writing the DR register, or by hardware when PE=0

23.6.7 I²C Status register 2 (I²C_SR2)

Address offset: 0x18

Reset value: 0x0000

Note: *Reading I²C_SR2 after reading I²C_SR1 clears the ADDR flag, even if the ADDR flag was set after reading I²C_SR1. Consequently, I²C_SR2 must be read only when ADDR is found set in I²C_SR1 or when the STOPF bit is cleared.*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
								PEC[7:0]	DUALF	SMB HOST	SMBDE FAULT	GEN CALL	Res.	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 15:8 **PEC[7:0]** Packet error checking register

This register contains the internal PEC when ENPEC=1.

Bit 7 **DUALF**: Dual flag (Slave mode)

- 0: Received address matched with OAR1
- 1: Received address matched with OAR2

–Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 6 **SMBHOST**: SMBus host header (Slave mode)

- 0: No SMBus Host address
- 1: SMBus Host address received when SMBTYPE=1 and ENARP=1.

–Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 5 **SMBDEFAULT**: SMBus device default address (Slave mode)

- 0: No SMBus Device Default address
- 1: SMBus Device Default address received when ENARP=1

–Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 4 **GENCALL**: General call address (Slave mode)

- 0: No General Call
- 1: General Call Address received when ENGC=1

–Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 3 Reserved, must be kept at reset value

Bit 2 **TRA**: Transmitter/receiver

- 0: Data bytes received
- 1: Data bytes transmitted

This bit is set depending on the R/W bit of the address byte, at the end of total address phase.

It is also cleared by hardware after detection of Stop condition (STOPF=1), repeated Start condition, loss of bus arbitration (ARLO=1), or when PE=0.

Bit 1 **BUSY**: Bus busy

- 0: No communication on the bus
- 1: Communication ongoing on the bus

–Set by hardware on detection of SDA or SCL low
–cleared by hardware on detection of a Stop condition.

It indicates a communication in progress on the bus. This information is still updated when the interface is disabled (PE=0).

Bit 0 **MSL:** Master/slave

- 0: Slave Mode
- 1: Master Mode

- Set by hardware as soon as the interface is in Master mode (SB=1).
- Cleared by hardware after detecting a Stop condition on the bus or a loss of arbitration (ARLO=1), or by hardware when PE=0.

Note: *Reading I²C_SR2 after reading I²C_SR1 clears the ADDR flag, even if the ADDR flag was set after reading I²C_SR1. Consequently, I²C_SR2 must be read only when ADDR is found set in I²C_SR1 or when the STOPF bit is cleared.*

23.6.8 I²C Clock control register (I²C_CCR)

Address offset: 0x1C

Reset value: 0x0000

Note: 1 *f_{PCLK1} must be at least 2 MHz to achieve standard mode I²C frequencies. It must be at least 4 MHz to achieve fast mode I²C frequencies. It must be a multiple of 10MHz to reach the 400 kHz maximum I²C fast mode clock.*
 2 *The CCR register must be configured only when the I²C is disabled (PE = 0).*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
F/S	DUTY	Reserved		CCR[11:0]													
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit 15 **F/S:** I²C master mode selection

- 0: Standard Mode I²C
- 1: Fast Mode I²C

Bit 14 **DUTY:** Fast mode duty cycle

- 0: Fast Mode t_{low}/t_{high} = 2
- 1: Fast Mode t_{low}/t_{high} = 16/9 (see CCR)

Bits 13:12 Reserved, must be kept at reset value

Bits 11:0 **CCR[11:0]**: Clock control register in Fast/Standard mode (Master mode)

Controls the SCL clock in master mode.

Standard mode or SMBus:

$$T_{\text{high}} = CCR * T_{\text{PCLK1}}$$

$$T_{\text{low}} = CCR * T_{\text{PCLK1}}$$

Fast mode:

If DUTY = 0:

$$T_{\text{high}} = CCR * T_{\text{PCLK1}}$$

$$T_{\text{low}} = 2 * CCR * T_{\text{PCLK1}}$$

If DUTY = 1: (to reach 400 kHz)

$$T_{\text{high}} = 9 * CCR * T_{\text{PCLK1}}$$

$$T_{\text{low}} = 16 * CCR * T_{\text{PCLK1}}$$

For instance: in standard mode, to generate a 100 kHz SCL frequency:

If FREQR = 08, $T_{\text{PCLK1}} = 125$ ns so CCR must be programmed with 0x28

(0x28 \leftrightarrow 40d \times 125 ns = 5000 ns.)

Note: 1. The minimum allowed value is 0x04, except in FAST DUTY mode where the minimum allowed value is 0x01
 . These timings are without filters.
 . The CCR register must be configured only when the I^2C is disabled (PE = 0).

23.6.9 I^2C TRISE register (I2C_TRISE)

Address offset: 0x20

Reset value: 0x0002

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

Bits 15:6 Reserved, must be kept at reset value

Bits 5:0 **TRISE[5:0]**: Maximum rise time in Fast/Standard mode (Master mode)

These bits must be programmed with the maximum SCL rise time given in the I^2C bus specification, incremented by 1.

For instance: in standard mode, the maximum allowed SCL rise time is 1000 ns.

If, in the I2C_CR2 register, the value of FREQ[5:0] bits is equal to 0x08 and $T_{\text{PCLK1}} = 125$ ns therefore the TRISE[5:0] bits must be programmed with 09h.

(1000 ns / 125 ns = 8 + 1)

The filter value can also be added to TRISE[5:0].

If the result is not an integer, TRISE[5:0] must be programmed with the integer part, in order to respect the t_{HIGH} parameter.

Note: TRISE[5:0] must be configured only when the I^2C is disabled (PE = 0).

23.6.10 I²C register map

The table below provides the I²C register map and reset values.

Table 81. I²C register map and reset values

Refer to [Table 1 on page 50](#) for the register boundary addresses table.

24 Universal synchronous asynchronous receiver transmitter (USART)

24.1 USART introduction

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The USART offers a very wide range of baud rates using a fractional baud rate generator.

It supports synchronous one-way communication and half-duplex single wire communication. It also supports the LIN (local interconnection network), Smartcard Protocol and IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). It allows multiprocessor communication.

High speed data communication is possible by using the DMA for multibuffer configuration.

24.2 USART main features

- Full duplex, asynchronous communications
- NRZ standard format (Mark/Space)
- Configurable oversampling method by 16 or by 8 to give flexibility between speed and clock tolerance
- Fractional baud rate generator systems
 - Common programmable transmit and receive baud rate (refer to the datasheets for the value of the baud rate at the maximum APB frequency).
- Programmable data word length (8 or 9 bits)
- Configurable stop bits - support for 1 or 2 stop bits
- LIN Master Synchronous Break send capability and LIN slave break detection capability
 - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- Transmitter clock output for synchronous transmission
- IrDA SIR encoder decoder
 - Support for 3/16 bit duration for normal mode
- Smartcard emulation capability
 - The Smartcard interface supports the asynchronous protocol Smartcards as defined in the ISO 7816-3 standards
 - 0.5, 1.5 stop bits for Smartcard operation
- Single-wire half-duplex communication
- Configurable multibuffer communication using DMA (direct memory access)
 - Buffering of received/transmitted bytes in reserved SRAM using centralized DMA
- Separate enable bits for transmitter and receiver

- Transfer detection flags:
 - Receive buffer full
 - Transmit buffer empty
 - End of transmission flags
- Parity control:
 - Transmits parity bit
 - Checks parity of received data byte
- Four error detection flags:
 - Overrun error
 - Noise detection
 - Frame error
 - Parity error
- Ten interrupt sources with flags:
 - CTS changes
 - LIN break detection
 - Transmit data register empty
 - Transmission complete
 - Receive data register full
 - Idle line received
 - Overrun error
 - Framing error
 - Noise error
 - Parity error
- Multiprocessor communication - enter into mute mode if address match does not occur
- Wake up from mute mode (by idle line detection or address mark detection)
- Two receiver wakeup modes: Address bit (MSB, 9th bit), Idle line

24.3 USART functional description

The interface is externally connected to another device by three pins (see [Figure 216](#)). Any USART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

RX: Receive Data Input is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.

TX: Transmit Data Output. When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire and smartcard modes, this I/O is used to transmit and receive the data (at USART level, data are then received on SW_RX).

Through these pins, serial data is transmitted and received in normal USART mode as frames comprising:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (8 or 9 bits) least significant bit first
- 0.5, 1, 1.5, 2 Stop bits indicating that the frame is complete
- This interface uses a fractional baud rate generator - with a 12-bit mantissa and 4-bit fraction
- A status register (USART_SR)
- Data Register (USART_DR)
- A baud rate register (USART_BRR) - 12-bit mantissa and 4-bit fraction.
- A Guardtime Register (USART_GTPR) in case of Smartcard mode.

Refer to [Section 24.6: USART registers on page 646](#) for the definitions of each bit.

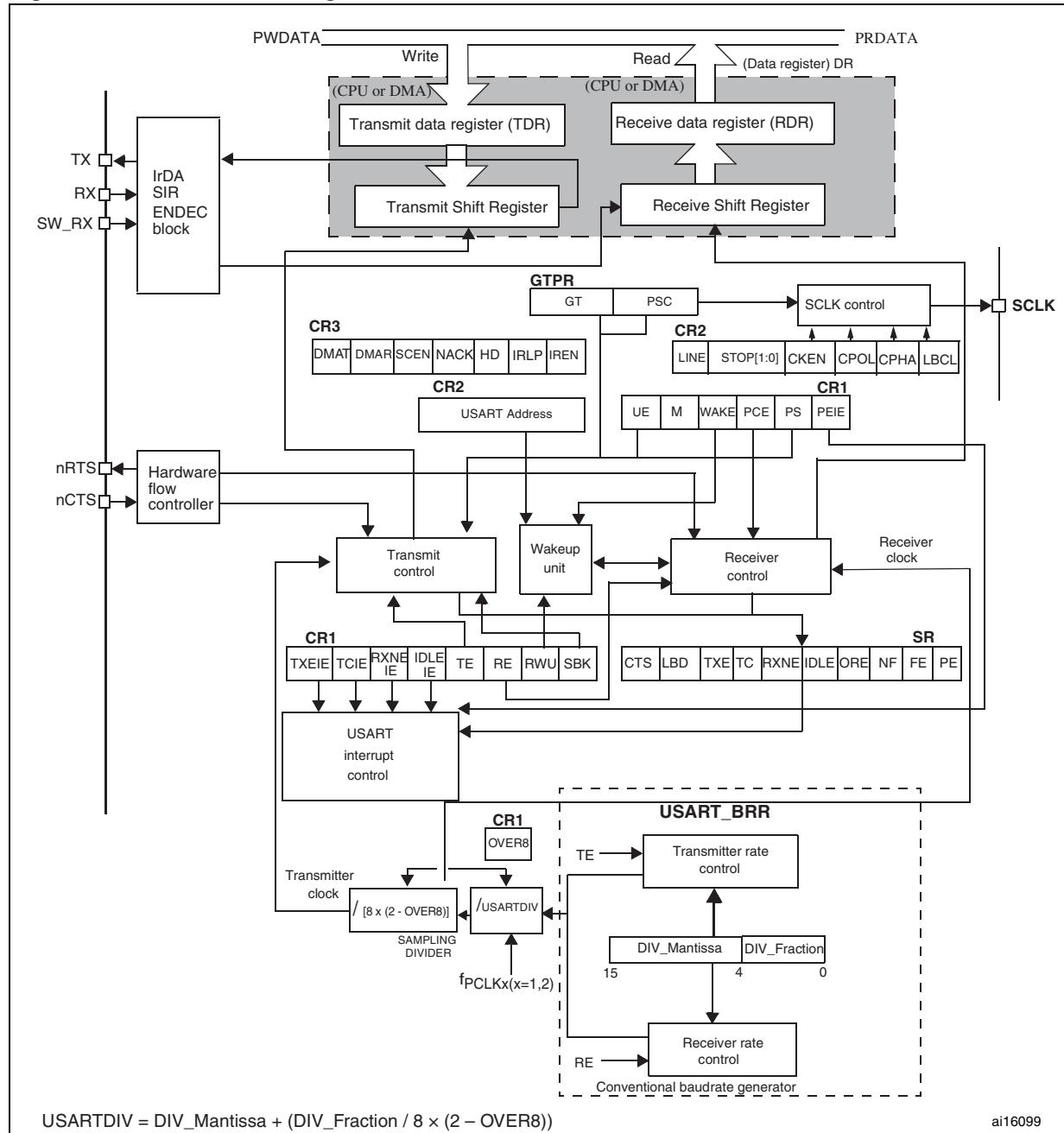
The following pin is required to interface in synchronous mode:

- **SCLK:** Transmitter clock output. This pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel data can be received synchronously on RX. This can be used to control peripherals that have shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable. In smartcard mode, SCLK can provide the clock to the smartcard.

The following pins are required in Hardware flow control mode:

- **nCTS:** Clear To Send blocks the data transmission at the end of the current transfer when high
- **nRTS:** Request to send indicates that the USART is ready to receive a data (when low).

Figure 216. USART block diagram



$$\text{USARTDIV} = \text{DIV_Mantissa} + (\text{DIV_Fraction} / 8 \times (2 - \text{OVER8}))$$

ai16099

24.3.1 USART character description

Word length may be selected as being either 8 or 9 bits by programming the M bit in the USART_CR1 register (see [Figure 217](#)).

The TX pin is in low state during the start bit. It is in high state during the stop bit.

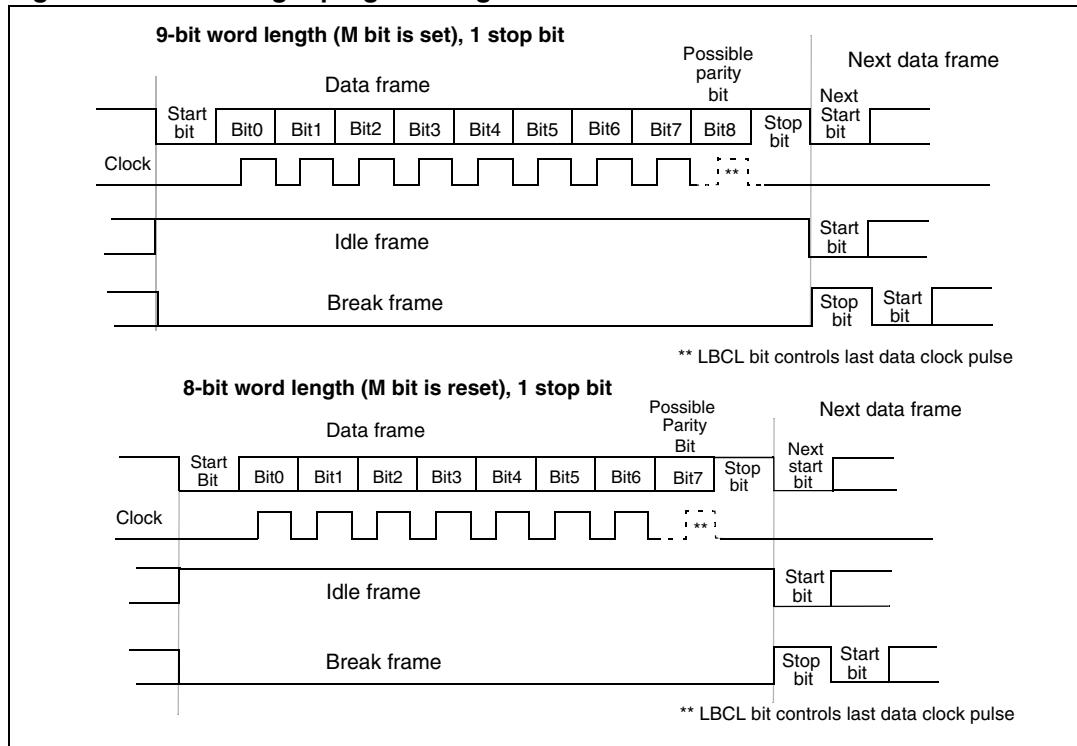
An ***Idle character*** is interpreted as an entire frame of “1”s followed by the start bit of the next frame which contains data (The number of “1” ‘s will include the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame the transmitter inserts either 1 or 2 stop bits (logic “1” bit) to acknowledge the start bit.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.

The details of each block is given below.

Figure 217. Word length programming



24.3.2 Transmitter

The transmitter can send data words of either 8 or 9 bits depending on the M bit status. When the transmit enable bit (TE) is set, the data in the transmit shift register is output on the TX pin and the corresponding clock pulses are output on the SCLK pin.

Character transmission

During an USART transmission, data shifts out least significant bit first on the TX pin. In this mode, the USART_DR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see [Figure 216](#)).

Every character is preceded by a start bit which is a logic level low for one bit period. The character is terminated by a configurable number of stop bits.

The following stop bits are supported by USART: 0.5, 1, 1.5 and 2 stop bits.

- Note:
- 1 *The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission will corrupt the data on the TX pin as the baud rate counters will get frozen. The current data being transmitted will be lost.*
 - 2 *An idle frame will be sent after the TE bit is enabled.*

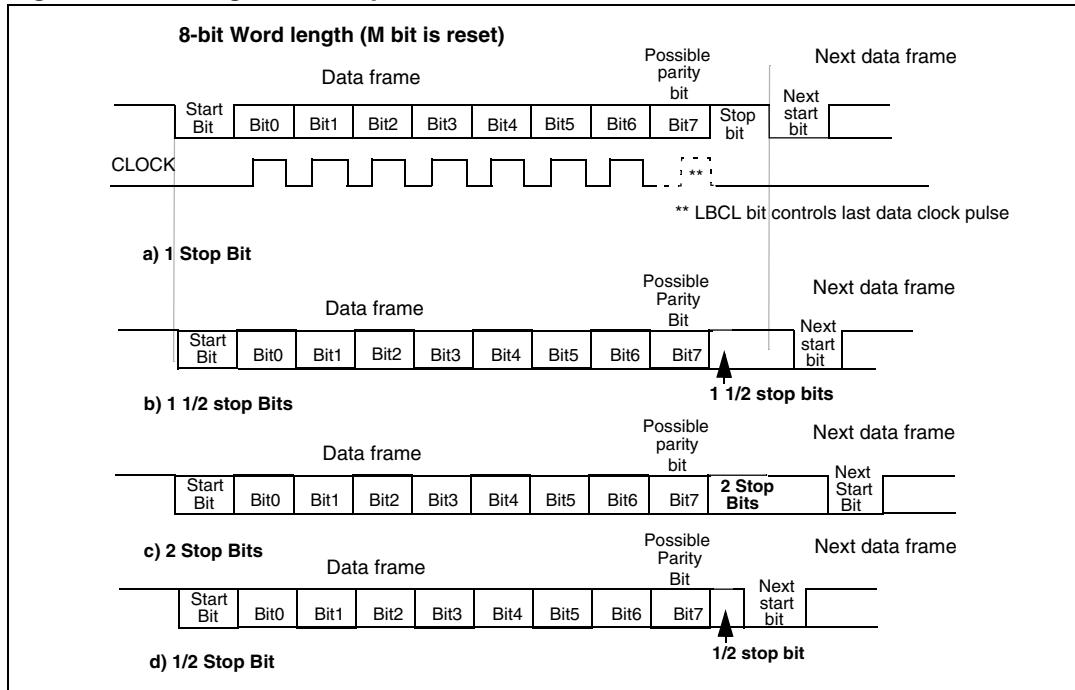
Configurable stop bits

The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.

- **1 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.
- **0.5 stop bit:** To be used when receiving data in Smartcard mode.
- **1.5 stop bits:** To be used when transmitting and receiving data in Smartcard mode.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits followed by the configured number of stop bits (when m = 0) and 11 low bits followed by the configured number of stop bits (when m = 1). It is not possible to transmit long breaks (break of length greater than 10/11 low bits).

Figure 218. Configurable stop bits**Procedure:**

1. Enable the USART by writing the UE bit in USART_CR1 register to 1.
2. Program the M bit in USART_CR1 to define the word length.
3. Program the number of stop bits in USART_CR2.
4. Select DMA enable (DMAT) in USART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
5. Select the desired baud rate using the USART_BRR register.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_DR register (this clears the TXE bit). Repeat this for each data to be transmitted in case of single buffer.
8. After writing the last data into the USART_DR register, wait until TC=1. This indicates that the transmission of the last frame is complete. This is required for instance when the USART is disabled or enters the Halt mode to avoid corrupting the last transmission.

Single byte communication

Clearing the TXE bit is always performed by a write to the data register.

The TXE bit is set by hardware and it indicates:

- The data has been moved from TDR to the shift register and the data transmission has started.
- The TDR register is empty.
- The next data can be written in the USART_DR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is taking place, a write instruction to the USART_DR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the USART_DR register places the data directly in the shift register, the data transmission starts, and the TXE bit is immediately set.

If a frame is transmitted (after the stop bit) and the TXE bit is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

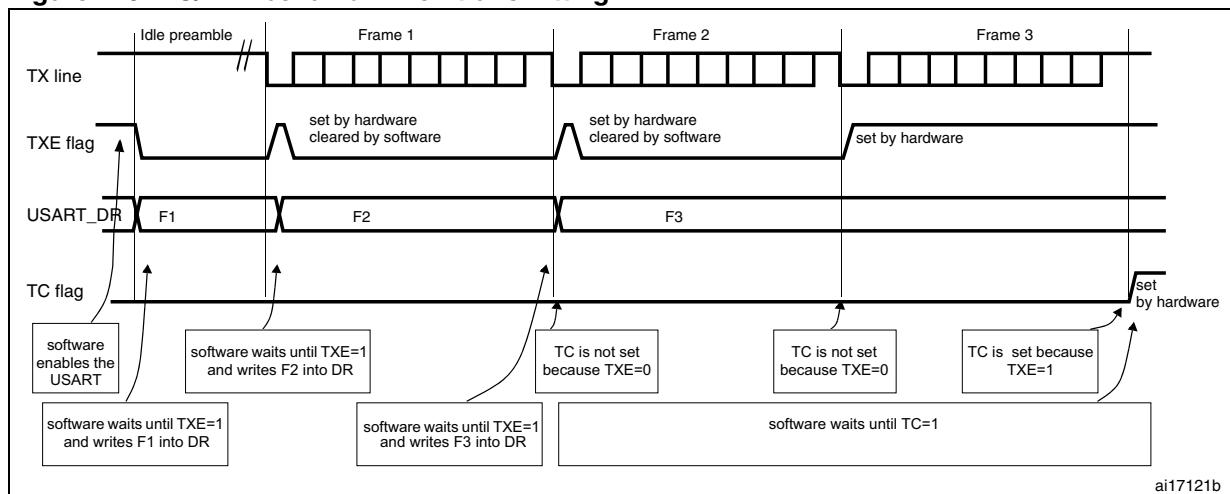
After writing the last data into the USART_DR register, it is mandatory to wait for TC=1 before disabling the USART or causing the microcontroller to enter the low power mode (see [Figure 219: TC/TXE behavior when transmitting](#)).

The TC bit is cleared by the following software sequence:

1. A read from the USART_SR register
2. A write to the USART_DR register

Note: *The TC bit can also be cleared by writing a '0' to it. This clearing sequence is recommended only for Multibuffer communication.*

Figure 219. TC/TXE behavior when transmitting



Break characters

Setting the SBK bit transmits a break character. The break frame length depends on the M bit (see [Figure 217](#)).

If the SBK bit is set to '1' a break character is sent on the TX line after completing the current character transmission. This bit is reset by hardware when the break character is completed (during the stop bit of the break character). The USART inserts a logic 1 bit at the end of the last break frame to guarantee the recognition of the start bit of the next frame.

Note: *If the software resets the SBK bit before the commencement of break transmission, the break character will not be transmitted. For two consecutive breaks, the SBK bit should be set after the stop bit of the previous break.*

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

24.3.3 Receiver

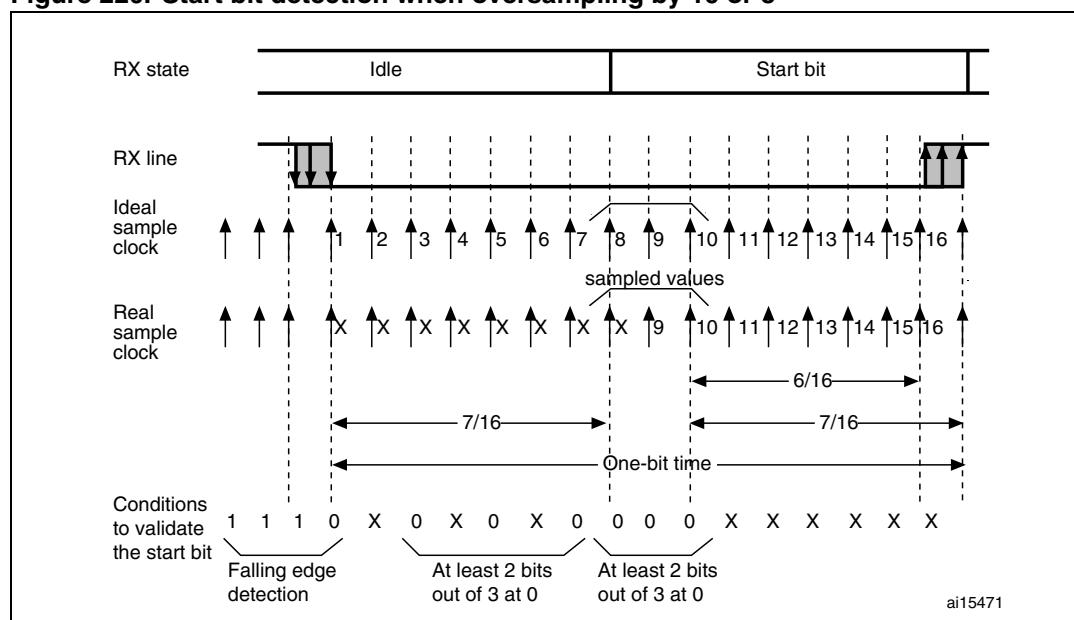
The USART can receive data words of either 8 or 9 bits depending on the M bit in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0 X 0 0 0 0.

Figure 220. Start bit detection when oversampling by 16 or 8



Note:

If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set) where it waits for a falling edge.

The start bit is confirmed (RXNE flag set, interrupt generated if RXNEIE=1) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated (RXNE flag set, interrupt generated if RXNEIE=1) but the NE noise flag is set if, for both samplings, at least 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits). If this condition is not met, the start detection aborts and the receiver returns to the idle state (no flag is set).

If, for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0, the start bit is validated but the NE noise flag bit is set.

Character reception

During an USART reception, data shifts in least significant bit first through the RX pin. In this mode, the USART_DR register consists of a buffer (RDR) between the internal bus and the received shift register.

Procedure:

1. Enable the USART by writing the UE bit in USART_CR1 register to 1.
2. Program the M bit in USART_CR1 to define the word length.
3. Program the number of stop bits in USART_CR2.
4. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication. STEP 3
5. Select the desired baud rate using the baud rate register USART_BRR
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- The RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- An interrupt is generated if the RXNEIE bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer, RXNE is set after every byte received and is cleared by the DMA read to the Data Register.
- In single buffer mode, clearing the RXNE bit is performed by a software read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.

Note:

The RE bit should not be reset while receiving data. If the RE bit is disabled during reception, the reception of the current byte will be aborted.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, there is the same procedure as a data received character plus an interrupt if the IDLEIE bit is set.

Overrun error

An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared.

The RXNE flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

- The ORE bit is set.

- The RDR content will not be lost. The previous data is available when a read to USART_DR is performed.
- The shift register will be overwritten. After that point, any data received during overrun is lost.
- An interrupt is generated if either the RXNEIE bit is set or both the EIE and DMAR bits are set.
- The ORE bit is reset by a read to the USART_SR register followed by a USART_DR register read operation.

Note: *The ORE bit, when set, indicates that at least 1 data has been lost. There are two possibilities:*

- *if RXNE=1, then the last valid data is stored in the receive register RDR and can be read,*
- *if RXNE=0, then it means that the last valid data has already been read and thus there is nothing to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received. It may also occur when the new data is received during the reading sequence (between the USART_SR register read access and the USART_DR read access).*

Selecting the proper oversampling method

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register and can be either 16 or 8 times the baud rate clock ([Figure 221](#) and [Figure 222](#)).

Depending on the application:

- select oversampling by 8 (OVER8=1) to achieve higher speed (up to $f_{PCLK}/8$). In this case the maximum receiver tolerance to clock deviation is reduced (refer to [Section 24.3.5: USART receiver tolerance to clock deviation on page 628](#))
- 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 $f_{PCLK}/16$

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. There are two options:

- the majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NF bit is set
- a single sample in the center of the received bit

Depending on the application:

- select the three samples' majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to [Figure 82](#) 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 24.3.5: USART receiver tolerance to clock deviation on page 628](#)). In this case the NF bit will never be set.

When noise is detected in a frame:

- The NF bit is set at the rising edge of the RXNE bit.
- The invalid data is transferred from the Shift register to the USART_DR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.

The NF bit is reset by a USART_SR register read operation followed by a USART_DR register read operation.

Note: *Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to '0' by hardware.*

Figure 221. Data sampling when oversampling by 16

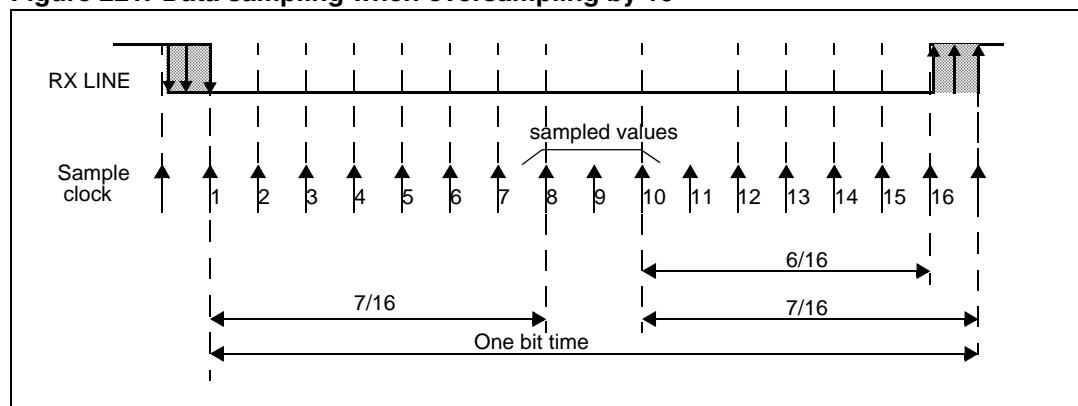


Figure 222. Data sampling when oversampling by 8

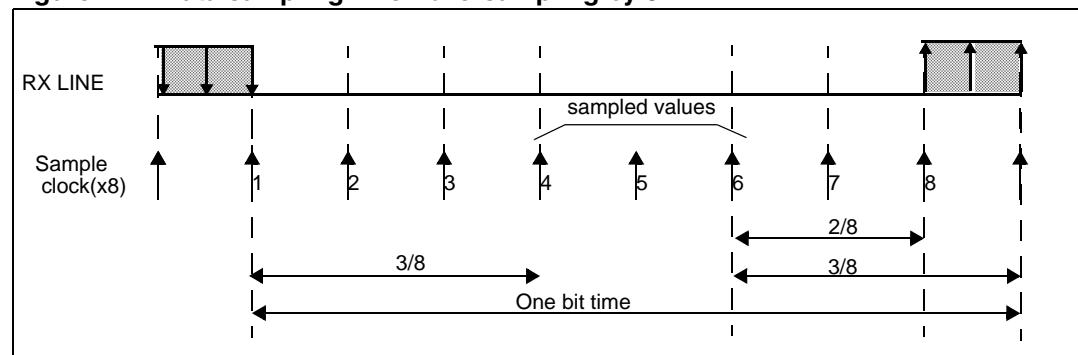


Table 82. Noise detection from sampled data

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

Table 82. Noise detection from sampled data (continued)

Sampled value	NE status	Received bit value
110	1	1
111	0	1

Framing error

A framing error is detected when:

The stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware
- The invalid data is transferred from the Shift register to the USART_DR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by a USART_SR register read operation followed by a USART_DR register read operation.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of Control Register 2 - it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

1. **0.5 stop bit (reception in Smartcard mode):** No sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
2. **1 stop bit:** Sampling for 1 stop Bit is done on the 8th, 9th and 10th samples.
3. **1.5 stop bits (Smartcard mode):** When transmitting in smartcard mode, the device must check that the data is correctly sent. Thus the receiver block must be enabled (RE =1 in the USART_CR1 register) and the stop bit is checked to test if the smartcard has detected a parity error. In the event of a parity error, the smartcard forces the data signal low during the sampling - NACK signal-, which is flagged as a framing error. Then, the FE flag is set with the RXNE at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be decomposed into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through. Refer to [Section 24.3.11: Smartcard on page 637](#) for more details.
4. **2 stop bits:** Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. If a framing error is detected during the first stop bit the framing error flag will be set. The second stop bit is not checked for framing error. The RXNE flag will be set at the end of the first stop bit.

24.3.4 Fractional baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the Mantissa and Fraction values of USARTDIV.

Equation 1: Baud rate for standard USART (SPI mode included)

$$\text{Tx/Rx baud} = \frac{f_{CK}}{8 \times (2 - \text{OVER8}) \times \text{USARTDIV}}$$

Equation 2: Baud rate in Smartcard, LIN and IrDA modes

$$\text{Tx/Rx baud} = \frac{f_{CK}}{16 \times \text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8=0, the fractional part is coded on 4 bits and programmed by the DIV_fraction[3:0] bits in the USART_BRR register
- When OVER8=1, the fractional part is coded on 3 bits and programmed by the DIV_fraction[2:0] bits in the USART_BRR register, and bit DIV_fraction[3] must be kept cleared.

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value should not be changed during communication.

How to derive USARTDIV from USART_BRR register values when OVER8=0

Example 1:

If DIV_Mantissa = 0d27 and DIV_Fraction = 0d12 (USART_BRR = 0x1BC), then

Mantissa (USARTDIV) = 0d27

Fraction (USARTDIV) = 12/16 = 0d0.75

Therefore USARTDIV = 0d27.75

Example 2:

To program USARTDIV = 0d25.62

This leads to:

DIV_Fraction = 16 * 0d0.62 = 0d9.92

The nearest real number is 0d10 = 0xA

DIV_Mantissa = mantissa (0d25.620) = 0d25 = 0x19

Then, USART_BRR = 0x19A hence USARTDIV = 0d25.625

Example 3:

To program USARTDIV = 0d50.99

This leads to:

DIV_Fraction = 16 * 0d0.99 = 0d15.84

The nearest real number is 0d16 = 0x10 => overflow of DIV_fra[3:0] => carry must be added up to the mantissa

DIV_Mantissa = mantissa (0d50.990 + carry) = 0d51 = 0x33

Then, USART_BRR = 0x330 hence USARTDIV = 0d51.000

How to derive USARTDIV from USART_BRR register values when OVER8=1

Example 1:

If DIV_Mantissa = 0x27 and DIV_Fraction[2:0]= 0d6 (USART_BRR = 0x1B6), then

Mantissa (USARTDIV) = 0d27

Fraction (USARTDIV) = 6/8 = 0d0.75

Therefore USARTDIV = 0d27.75

Example 2:

To program USARTDIV = 0d25.62

This leads to:

DIV_Fraction = 8*0d0.62 = 0d4.96

The nearest real number is 0d5 = 0x5

DIV_Mantissa = mantissa (0d25.620) = 0d25 = 0x19

Then, USART_BRR = 0x195 => USARTDIV = 0d25.625

Example 3:

To program USARTDIV = 0d50.99

This leads to:

DIV_Fraction = 8*0d0.99 = 0d7.92

The nearest real number is 0d8 = 0x8 => overflow of the DIV_fra[2:0] => carry must be added up to the mantissa

DIV_Mantissa = mantissa (0d50.990 + carry) = 0d51 = 0x33

Then, USART_BRR = 0x0330 => USARTDIV = 0d51.000

Table 83. Error calculation for programmed baud rates at $f_{PCLK} = 8 \text{ MHz}$ or $f_{PCLK} = 12 \text{ MHz}$, oversampling by 16⁽¹⁾

Oversampling by 16 (OVER8=0)							
Baud rate7		$f_{PCLK} = 8 \text{ MHz}$			$f_{PCLK} = 12 \text{ MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	416.6875	0	1.2 KBps	625	0
2	2.4 KBps	2.4 KBps	208.3125	0.01	2.4 KBps	312.5	0
3	9.6 KBps	9.604 KBps	52.0625	0.04	9.6 KBps	78.125	0
4	19.2 KBps	19.185 KBps	26.0625	0.08	19.2 KBps	39.0625	0

Table 83. Error calculation for programmed baud rates at $f_{PCLK} = 8 \text{ MHz}$ or $f_{PCLK} = 12 \text{ MHz}$, oversampling by 16⁽¹⁾ (continued)

Oversampling by 16 (OVER8=0)							
Baud rate7		$f_{PCLK} = 8 \text{ MHz}$			$f_{PCLK} = 12 \text{ MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
5	38.4 KBps	38.462 KBps	13	0.16	38.339 KBps	19.5625	0.16
6	57.6 KBps	57.554 KBps	8.6875	0.08	57.692 KBps	13	0.16
7	115.2 KBps	115.942 KBps	4.3125	0.64	115.385 KBps	6.5	0.16
8	230.4 KBps	228.571 KBps	2.1875	0.79	230.769 KBps	3.25	0.16
9	460.8 KBps	470.588 KBps	1.0625	2.12	461.538 KBps	1.625	0.16
10	921.6 KBps	NA	NA	NA	NA	NA	NA
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

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.

Table 84. Error calculation for programmed baud rates at $f_{PCLK} = 8 \text{ MHz}$ or $f_{PCLK} = 12 \text{ MHz}$, oversampling by 8⁽¹⁾

Oversampling by 8 (OVER8 = 1)							
Baud rate		$f_{PCLK} = 8 \text{ MHz}$			$f_{PCLK} = 12 \text{ MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	833.375	0	1.2 KBps	1250	0
2	2.4 KBps	2.4 KBps	416.625	0.01	2.4 KBps	625	0
3	9.6 KBps	9.604 KBps	104.125	0.04	9.6 KBps	156.25	0
4	19.2 KBps	19.185 KBps	52.125	0.08	19.2 KBps	78.125	0
5	38.4 KBps	38.462 KBps	26	0.16	38.339 KBps	39.125	0.16
6	57.6 KBps	57.554 KBps	17.375	0.08	57.692 KBps	26	0.16
7	115.2 KBps	115.942 KBps	8.625	0.64	115.385 KBps	13	0.16
8	230.4 KBps	228.571 KBps	4.375	0.79	230.769 KBps	6.5	0.16
9	460.8 KBps	470.588 KBps	2.125	2.12	461.538 KBps	3.25	0.16
10	921.6 KBps	888.889 KBps	1.125	3.55	923.077 KBps	1.625	0.16

Table 84. Error calculation for programmed baud rates at $f_{PCLK} = 8 \text{ MHz}$ or $f_{PCLK} = 12 \text{ MHz}$, oversampling by 8⁽¹⁾ (continued)

Oversampling by 8 (OVER8 = 1)							
Baud rate		$f_{PCLK} = 8 \text{ MHz}$			$f_{PCLK} = 12 \text{ MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

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.

Table 85. Error calculation for programmed baud rates at $f_{PCLK} = 16 \text{ MHz}$ or $f_{PCLK} = 24 \text{ MHz}$, oversampling by 16⁽¹⁾

Oversampling by 16 (OVER8 = 0)							
Baud rate		$f_{PCLK} = 16 \text{ MHz}$			$f_{PCLK} = 24 \text{ MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	833.3125	0	1.2	1250	0
2	2.4 KBps	2.4 KBps	416.6875	0	2.4	625	0
3	9.6 KBps	9.598 KBps	104.1875	0.02	9.6	156.25	0
4	19.2 KBps	19.208 KBps	52.0625	0.04	19.2	78.125	0
5	38.4 KBps	38.369 KBps	26.0625	0.08	38.4	39.0625	0
6	57.6 KBps	57.554 KBps	17.375	0.08	57.554	26.0625	0.08
7	115.2 KBps	115.108 KBps	8.6875	0.08	115.385	13	0.16
8	230.4 KBps	231.884 KBps	4.3125	0.64	230.769	6.5	0.16
9	460.8 KBps	457.143 KBps	2.1875	0.79	461.538	3.25	0.16
10	921.6 KBps	941.176 KBps	1.0625	2.12	923.077	1.625	0.16
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

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.

Table 86. Error calculation for programmed baud rates at $f_{PCLK} = 16\text{ MHz}$ or $f_{PCLK} = 24\text{ MHz}$, oversampling by 8⁽¹⁾

Oversampling by 8 (OVER8=1)							
Baud rate		$f_{PCLK} = 16\text{ MHz}$			$f_{PCLK} = 24\text{ MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	1666.625	0	1.2 KBps	2500	0
2	2.4 KBps	2.4 KBps	833.375	0	2.4 KBps	1250	0
3	9.6 KBps	9.598 KBps	208.375	0.02	9.6 KBps	312.5	0
4	19.2 KBps	19.208 KBps	104.125	0.04	19.2 KBps	156.25	0
5	38.4 KBps	38.369 KBps	52.125	0.08	38.4 KBps	78.125	0
6	57.6 KBps	57.554 KBps	34.75	0.08	57.554 KBps	52.125	0.08
7	115.2 KBps	115.108 KBps	17.375	0.08	115.385 KBps	26	0.16
8	230.4 KBps	231.884 KBps	8.625	0.64	230.769 KBps	13	0.16
9	460.8 KBps	457.143 KBps	4.375	0.79	461.538 KBps	6.5	0.16
10	921.6 KBps	941.176 KBps	2.125	2.12	923.077 KBps	3.25	0.16
11	2 MBps	2000 KBps	1	0	2000 KBps	1.5	0
12	3 MBps	NA	NA	NA	3000 KBps	1	0

- 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.

Table 87. Error calculation for programmed baud rates at $f_{PCLK} = 8\text{ MHz}$ or $f_{PCLK} = 16\text{ MHz}$, oversampling by 16⁽¹⁾

Oversampling by 16 (OVER8=0)							
Baud rate		$f_{PCLK} = 8\text{ MHz}$			$f_{PCLK} = 16\text{ MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1.	2.4 KBps	2.400 KBps	208.3125	0.00%	2.400 KBps	416.6875	0.00%
2.	9.6 KBps	9.604 KBps	52.0625	0.04%	9.598 KBps	104.1875	0.02%
3.	19.2 KBps	19.185 KBps	26.0625	0.08%	19.208 KBps	52.0625	0.04%
4.	57.6 KBps	57.554 KBps	8.6875	0.08%	57.554 KBps	17.3750	0.08%
5.	115.2 KBps	115.942 KBps	4.3125	0.64%	115.108 KBps	8.6875	0.08%
6.	230.4 KBps	228.571 KBps	2.1875	0.79%	231.884 KBps	4.3125	0.64%
7.	460.8 KBps	470.588 KBps	1.0625	2.12%	457.143 KBps	2.1875	0.79%
8.	896 KBps	NA	NA	NA	888.889 KBps	1.1250	0.79%

Table 87. Error calculation for programmed baud rates at $f_{PCLK} = 8 \text{ MHz}$ or $f_{PCLK} = 16 \text{ MHz}$, oversampling by 16⁽¹⁾ (continued)

Oversampling by 16 (OVER8=0)							
Baud rate		$f_{PCLK} = 8 \text{ MHz}$			$f_{PCLK} = 16 \text{ MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
9.	921.6 KBps	NA	NA	NA	941.176 KBps	1.0625	2.12%
10.	1.792 MBps	NA	NA	NA	NA	NA	NA
11.	1.8432 MBps	NA	NA	NA	NA	NA	NA
12.	3.584 MBps	NA	NA	NA	NA	NA	NA
13.	3.6864 MBps	NA	NA	NA	NA	NA	NA
14.	7.168 MBps	NA	NA	NA	NA	NA	NA
15.	7.3728 MBps	NA	NA	NA	NA	NA	NA

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.

Table 88. Error calculation for programmed baud rates at $f_{PCLK} = 8 \text{ MHz}$ or $f_{PCLK} = 16 \text{ MHz}$, oversampling by 8⁽¹⁾

Oversampling by 8 (OVER8=1)							
Baud rate		$f_{PCLK} = 8 \text{ MHz}$			$f_{PCLK} = 16 \text{ MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1.	2.4 KBps	2.400 KBps	416.625	0.01%	2.400 KBps	833.375	0.00%
2.	9.6 KBps	9.604 KBps	104.125	0.04%	9.598 KBps	208.375	0.02%
3.	19.2 KBps	19.185 KBps	52.125	0.08%	19.208 KBps	104.125	0.04%
4.	57.6 KBps	57.557 KBps	17.375	0.08%	57.554 KBps	34.750	0.08%
5.	115.2 KBps	115.942 KBps	8.625	0.64%	115.108 KBps	17.375	0.08%
6.	230.4 KBps	228.571 KBps	4.375	0.79%	231.884 KBps	8.625	0.64%
7.	460.8 KBps	470.588 KBps	2.125	2.12%	457.143 KBps	4.375	0.79%
8.	896 KBps	888.889 KBps	1.125	0.79%	888.889 KBps	2.250	0.79%
9.	921.6 KBps	888.889 KBps	1.125	3.55%	941.176 KBps	2.125	2.12%
10.	1.792 MBps	NA	NA	NA	1.7777 MBps	1.125	0.79%
11.	1.8432 MBps	NA	NA	NA	1.7777 MBps	1.125	3.55%
12.	3.584 MBps	NA	NA	NA	NA	NA	NA
13.	3.6864 MBps	NA	NA	NA	NA	NA	NA
14.	7.168 MBps	NA	NA	NA	NA	NA	NA
15.	7.3728 MBps	NA	NA	NA	NA	NA	NA

- 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.

Table 89. Error calculation for programmed baud rates at $f_{PCLK} = 30\text{ MHz}$ or $f_{PCLK} = 60\text{ MHz}$, oversampling by 16⁽¹⁾⁽²⁾

Oversampling by 16 (OVER8=0)							
Baud rate		$f_{PCLK} = 30\text{ MHz}$			$f_{PCLK} = 60\text{ MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1.	2.4 KBps	2.400 KBps	781.2500	0.00%	2.400 KBps	1562.5000	0.00%
2.	9.6 KBps	9.600 KBps	195.3125	0.00%	9.600 KBps	390.6250	0.00%
3.	19.2 KBps	19.194 KBps	97.6875	0.03%	19.200 KBps	195.3125	0.00%
4.	57.6 KBps	57.582 KBps	32.5625	0.03%	57.582 KBps	65.1250	0.03%
5.	115.2 KBps	115.385 KBps	16.2500	0.16%	115.163 KBps	32.5625	0.03%
6.	230.4 KBps	230.769 KBps	8.1250	0.16%	230.769 KBps	16.2500	0.16%
7.	460.8 KBps	461.538 KBps	4.0625	0.16%	461.538 KBps	8.1250	0.16%
8.	896 KBps	909.091 KBps	2.0625	1.46%	895.522 KBps	4.1875	0.05%
9.	921.6 KBps	909.091 KBps	2.0625	1.36%	923.077 KBps	4.0625	0.16%
10.	1.792 MBps	1.1764 MBps	1.0625	1.52%	1.8182 MBps	2.0625	1.36%
11.	1.8432 MBps	1.8750 MBps	1.0000	1.73%	1.8182 MBps	2.0625	1.52%
12.	3.584 MBps	NA	NA	NA	3.2594 MBps	1.0625	1.52%
13.	3.6864 MBps	NA	NA	NA	3.7500 MBps	1.0000	1.73%
14.	7.168 MBps	NA	NA	NA	NA	NA	NA
15.	7.3728 MBps	NA	NA	NA	NA	NA	NA

- 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.
- Only USART1 and USART6 are clocked with PCLK2. Other USARTs are clocked with PCLK1. Refer to the device datasheets for the maximum values for PCLK1 and PCLK2.

Table 90. Error calculation for programmed baud rates at $f_{PCLK} = 30\text{ MHz}$ or $f_{PCLK} = 60\text{ MHz}$, oversampling by 8⁽¹⁾⁽²⁾

Oversampling by 8 (OVER8=1)							
Baud rate		$f_{PCLK} = 30\text{ MHz}$			$f_{PCLK} = 60\text{ MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1.	2.4 KBps	2.400 KBps	1562.5000	0.00%	2.400 KBps	3125.0000	0.00%
2.	9.6 KBps	9.600 KBps	390.6250	0.00%	9.600 KBps	781.2500	0.00%

Table 90. Error calculation for programmed baud rates at $f_{PCLK} = 30$ MHz or $f_{PCLK} = 60$ MHz, oversampling by 8⁽¹⁾⁽²⁾ (continued)

Oversampling by 8 (OVER8=1)							
Baud rate		$f_{PCLK} = 30$ MHz			$f_{PCLK} = 60$ MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
3.	19.2 KBps	19.194 KBps	195.3750	0.03%	19.200 KBps	390.6250	0.00%
4.	57.6 KBps	57.582 KBps	65.1250	0.16%	57.582 KBps	130.2500	0.03%
5.	115.2 KBps	115.385 KBps	32.5000	0.16%	115.163 KBps	65.1250	0.03%
6.	230.4 KBps	230.769 KBps	16.2500	0.16%	230.769 KBps	32.5000	0.16%
7.	460.8 KBps	461.538 KBps	8.1250	0.16%	461.538 KBps	16.2500	0.16%
8.	896 KBps	909.091 KBps	4.1250	1.46%	895.522 KBps	8.3750	0.05%
9.	921.6 KBps	909.091 KBps	4.1250	1.36%	923.077 KBps	8.1250	0.16%
10.	1.792 MBps	1.7647 MBps	2.1250	1.52%	1.8182 MBps	4.1250	1.46%
11.	1.8432 MBps	1.8750 MBps	2.0000	1.73%	1.8182 MBps	4.1250	1.36%
12.	3.584 MBps	3.7500 MBps	1.0000	4.63%	3.5294 MBps	2.1250	1.52%
13.	3.6864 MBps	3.7500 MBps	1.0000	1.73%	3.7500 MBps	2.0000	1.73%
14.	7.168 MBps	NA	NA	NA	7.5000 MBps	1.0000	4.63%
15.	7.3728 MBps	NA	NA	NA	7.5000 MBps	1.0000	1.73%

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.
2. Only USART1 and USART6 are clocked with PCLK2. Other USARTs are clocked with PCLK1. Refer to the device datasheets for the maximum values for PCLK1 and PCLK2.

Table 91. Error calculation for programmed baud rates at $f_{PCLK} = 42$ MHz or $f_{PCLK} = 84$ Hz, oversampling by 16⁽¹⁾⁽²⁾

Oversampling by 16 (OVER8=0)							
Baud rate		$f_{PCLK} = 42$ MHz			$f_{PCLK} = 84$ MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1.	2.4 KBps	TBD	TBD	TBD	TBD	TBD	TBD
2.	9.6 KBps	TBD	TBD	TBD	TBD	TBD	TBD
3.	19.2 KBps	TBD	TBD	TBD	TBD	TBD	TBD
4.	57.6 KBps	TBD	TBD	TBD	TBD	TBD	TBD
5.	115.2 KBps	TBD	TBD	TBD	TBD	TBD	TBD
6.	230.4 KBps	TBD	TBD	TBD	TBD	TBD	TBD
7.	460.8 KBps	TBD	TBD	TBD	TBD	TBD	TBD

Table 91. Error calculation for programmed baud rates at $f_{PCLK} = 42$ MHz or $f_{PCLK} = 84$ Hz, oversampling by 16⁽¹⁾⁽²⁾ (continued)

Oversampling by 16 (OVER8=0)							
Baud rate		$f_{PCLK} = 42$ MHz			$f_{PCLK} = 84$ MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
8.	896 KBps	TBD	TBD	TBD	TBD	TBD	TBD
9.	921.6 KBps	TBD	TBD	TBD	TBD	TBD	TBD
10.	1.792 MBps	TBD	TBD	TBD	TBD	TBD	TBD
11.	1.8432 MBps	TBD	TBD	TBD	TBD	TBD	TBD
12.	3.584 MBps	TBD	TBD	TBD	TBD	TBD	TBD
13.	3.6864 MBps	TBD	TBD	TBD	TBD	TBD	TBD
14.	7.168 MBps	TBD	TBD	TBD	TBD	TBD	TBD
15.	7.3728 MBps	TBD	TBD	TBD	TBD	TBD	TBD

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.
2. Only USART1 and USART6 are clocked with PCLK2. Other USARTs are clocked with PCLK1. Refer to the device datasheets for the maximum values for PCLK1 and PCLK2.

Table 92. Error calculation for programmed baud rates at $f_{PCLK} = 42$ MHz or $f_{PCLK} = 84$ MHz, oversampling by 8⁽¹⁾⁽²⁾

Oversampling by 8 (OVER8=1)							
Baud rate		$f_{PCLK} = 42$ MHz			$f_{PCLK} = 84$ MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1.	2.4 KBps	TBD	TBD	TBD	TBD	TBD	TBD
2.	9.6 KBps	TBD	TBD	TBD	TBD	TBD	TBD
3.	19.2 KBps	TBD	TBD	TBD	TBD	TBD	TBD
4.	57.6 KBps	TBD	TBD	TBD	TBD	TBD	TBD
5.	115.2 KBps	TBD	TBD	TBD	TBD	TBD	TBD
6.	230.4 KBps	TBD	TBD	TBD	TBD	TBD	TBD
7.	460.8 KBps	TBD	TBD	TBD	TBD	TBD	TBD
8.	896 KBps	TBD	TBD	TBD	TBD	TBD	TBD
9.	921.6 KBps	TBD	TBD	TBD	TBD	TBD	TBD
10.	1.792 MBps	TBD	TBD	TBD	TBD	TBD	TBD
11.	1.8432 MBps	TBD	TBD	TBD	TBD	TBD	TBD
12.	3.584 MBps	TBD	TBD	TBD	TBD	TBD	TBD

Table 92. Error calculation for programmed baud rates at $f_{PCLK} = 42$ MHz or $f_{PCLK} = 84$ MHz, oversampling by 8⁽¹⁾⁽²⁾ (continued)

Oversampling by 8 (OVER8=1)							
Baud rate		$f_{PCLK} = 42$ MHz			$f_{PCLK} = 84$ MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
13.	3.6864 MBps	TBD	TBD	TBD	TBD	TBD	TBD
14.	7.168 MBps	TBD	TBD	TBD	TBD	TBD	TBD
15.	7.3728 MBps	TBD	TBD	TBD	TBD	TBD	TBD

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.
2. Only USART1 and USART6 are clocked with PCLK2. Other USARTs are clocked with PCLK1. Refer to the device datasheets for the maximum values for PCLK1 and PCLK2.

24.3.5 USART receiver tolerance to clock deviation

The USART's asynchronous receiver works correctly only if the total clock system deviation is smaller than the USART receiver's tolerance. The causes which contribute to the total deviation are:

- DTRA: Deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: Error due to the baud rate quantization of the receiver
- DREC: Deviation of the receiver's local oscillator
- DTCL: Deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL < \text{USART receiver's tolerance}$$

The USART receiver's tolerance to properly receive data is equal to the maximum tolerated deviation and depends on the following choices:

- 10- or 11-bit character length defined by the M bit in the USART_CR1 register
- oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- use of fractional baud rate or not
- use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register

Table 93. USART receiver's tolerance when DIV fraction is 0

M bit	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
0	3.75%	4.375%	2.50%	3.75%
1	3.41%	3.97%	2.27%	3.41%

Table 94. USART receiver's tolerance when DIV_Fraction is different from 0

M bit	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
0	3.33%	3.88%	2%	3%
1	3.03%	3.53%	1.82%	2.73%

Note: The figures specified in [Table](#) and [Table 94](#) may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M=0 (11-bit times when M=1).

24.3.6 Multiprocessor communication

There is a possibility of performing multiprocessor communication with the USART (several USARTs connected in a network). For instance one of the USARTs can be the master, its TX output is connected to the RX input of the other USART. The others are slaves, their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non addressed devices may be placed in mute mode by means of the muting function.
In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in USART_CR1 register is set to 1. RWU can be controlled automatically by hardware or written by the software under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

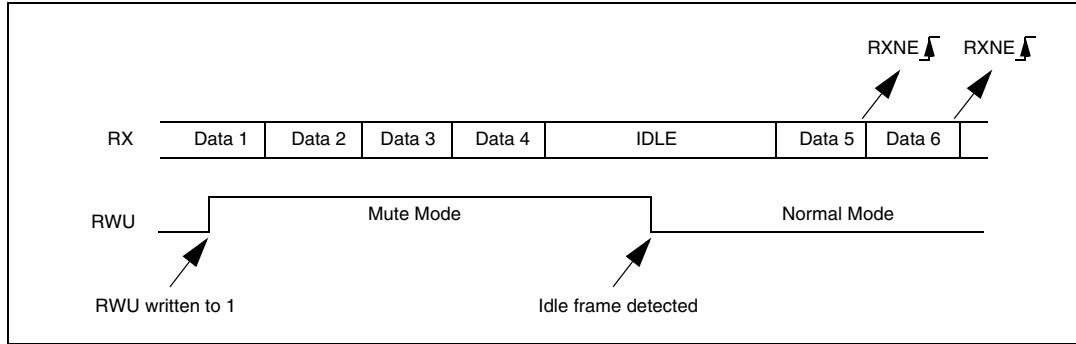
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The USART enters mute mode when the RWU bit is written to 1.

It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the USART_SR register. RWU can also be written to 0 by software.

An example of mute mode behavior using Idle line detection is given in [Figure 223](#).

Figure 223. Mute mode using Idle line detection**Address mark detection (WAKE=1)**

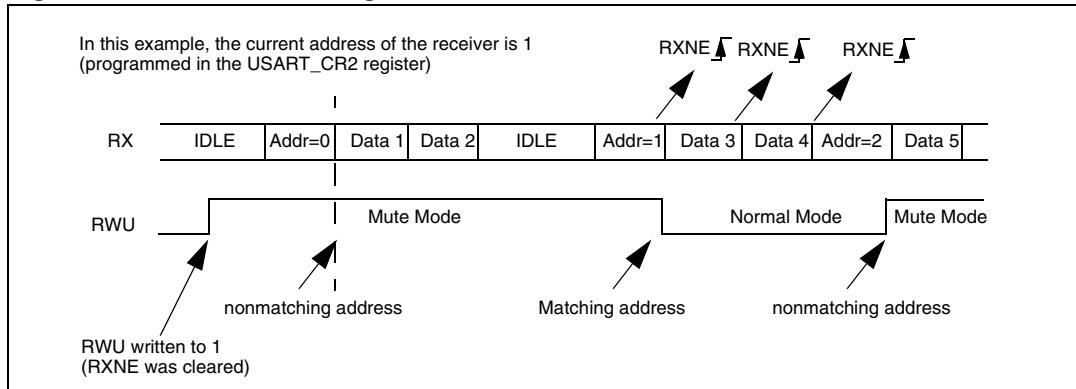
In this mode, bytes are recognized as addresses if their MSB is a '1' else they are considered as data. In an address byte, the address of the targeted receiver is put on the 4 LSB. This 4-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt nor DMA request is issued as the USART would have entered mute mode.

It exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE bit is set for the address character since the RWU bit has been cleared.

The RWU bit can be written to as 0 or 1 when the receiver buffer contains no data (RXNE=0 in the USART_SR register). Otherwise the write attempt is ignored.

An example of mute mode behavior using address mark detection is given in [Figure 224](#).

Figure 224. Mute mode using address mark detection

24.3.7 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bit, the possible USART frame formats are as listed in [Table 95](#).

Table 95. Frame formats

M bit	PCE bit	USART frame ⁽¹⁾
0	0	SB 8 bit data STB
0	1	SB 7-bit data PB STB
1	0	SB 9-bit data STB
1	1	SB 8-bit data PB STB

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

E.g.: data=00110101; 4 bits set => parity bit will be 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

E.g.: data=00110101; 4 bits set => parity bit will be 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_SR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by a software sequence (a read from the status register followed by a read or write access to the USART_DR data register).

Note:

In case of wakeup by an address mark: the MSB bit of the data is taken into account to identify an address but not the parity bit. And the receiver does not check the parity of the address data (PE is not set in case of a parity error).

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

Note:

The software routine that manages the transmission can activate the software sequence which clears the PE flag (a read from the status register followed by a read or write access to the data register). When operating in half-duplex mode, depending on the software, this can cause the PE flag to be unexpectedly cleared.

24.3.8 LIN (local interconnection network) mode

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- CLKEN in the USART_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The same procedure explained in [Section 24.3.2](#) has to be applied for LIN Master transmission than for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBK bit sends 13 '0' bits as a break character. Then a bit of value '1' is sent to allow the next start detection.

LIN reception

A break detection circuit is implemented on the USART interface. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL=1 in USART_CR2) consecutive bits are detected as '0', and are followed by a delimiter character, the LBD flag is set in USART_SR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a '1' is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at '0', which will be the case for any break frame), the receiver stops until the break detection circuit receives either a '1', if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the [Figure 225: Break detection in LIN mode \(11-bit break length - LBDL bit is set\) on page 633](#).

Examples of break frames are given on [Figure 226: Break detection in LIN mode vs. Framing error detection on page 634](#).

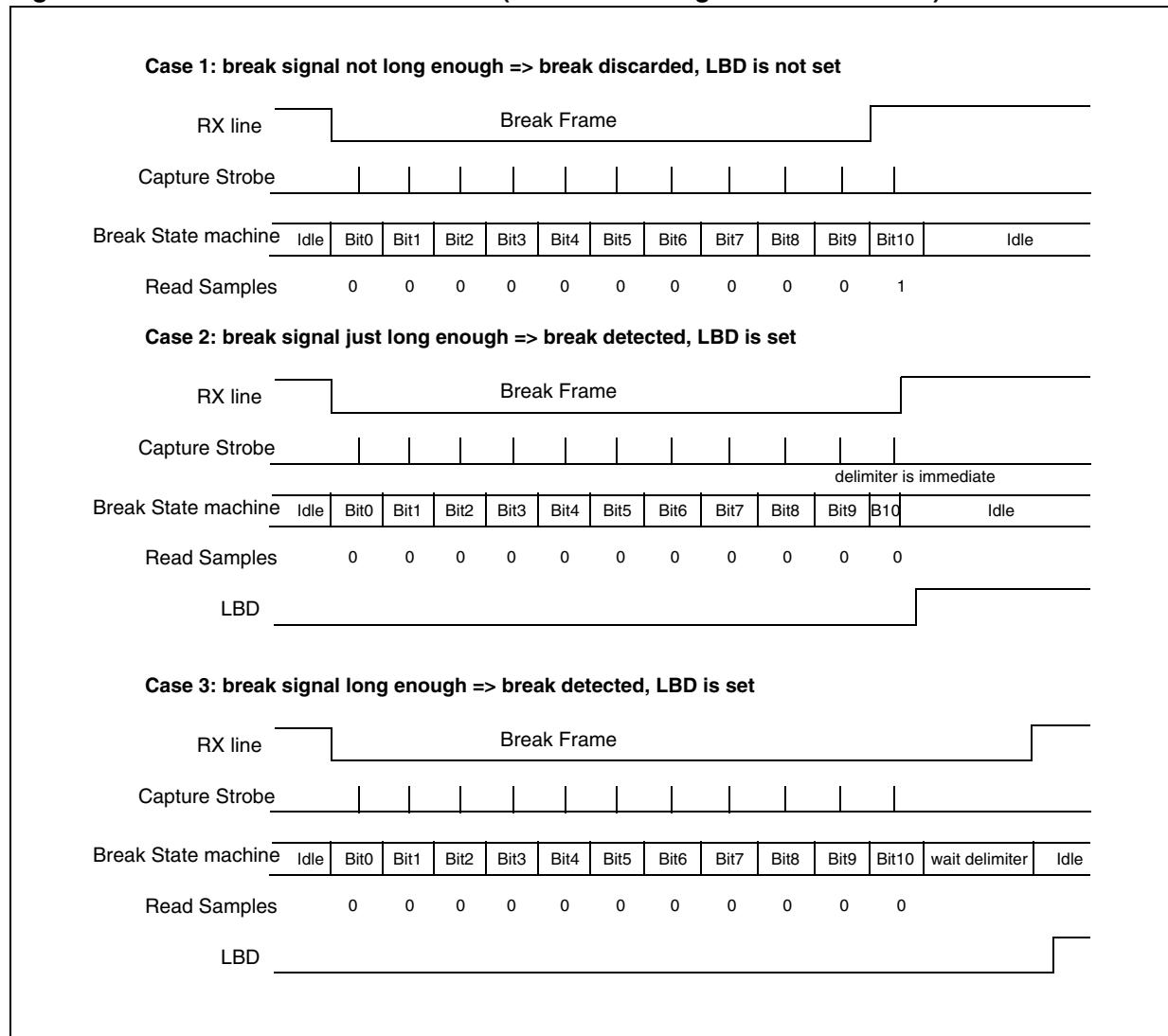
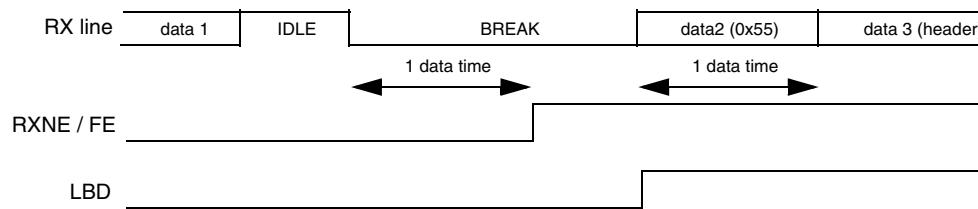
Figure 225. Break detection in LIN mode (11-bit break length - LBDL bit is set)

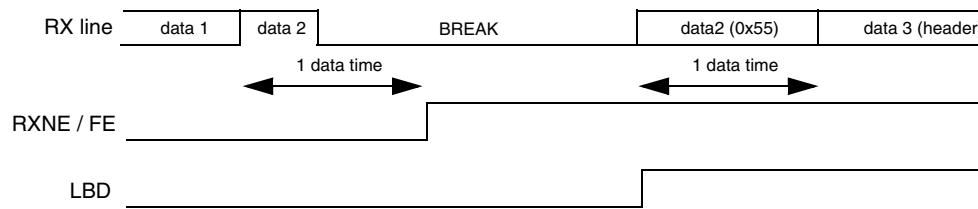
Figure 226. Break detection in LIN mode vs. Framing error detection

In these examples, we suppose that LBDL=1 (11-bit break length), M=0 (8-bit data)

Case 1: break occurring after an Idle



Case 1: break occurring while a data is being received



24.3.9 USART synchronous mode

The synchronous mode is selected by writing the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

The USART allows the user to control a bidirectional synchronous serial communications in master mode. The SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register clock pulses will or will not be generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register allows the user to select the clock polarity, and the CPHA bit in the USART_CR2 register allows the user to select the phase of the external clock (see [Figure 227](#), [Figure 228](#) & [Figure 229](#)).

During the Idle state, preamble and send break, the external SCLK clock is not activated.

In synchronous mode the USART transmitter works exactly like in asynchronous mode. But as SCLK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In this mode the USART receiver works in a different manner compared to the asynchronous mode. If RE=1, the data is sampled on SCLK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

- Note:*
- 1 *The SCLK pin works in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and a data is being transmitted (the data register USART_DR has been written). This means that it is not possible to receive a synchronous data without transmitting data.*
 - 2 *The LBCL, CPOL and CPHA bits have to be selected when both the transmitter and the receiver are disabled (TE=RE=0) to ensure that the clock pulses function correctly. These bits should not be changed while the transmitter or the receiver is enabled.*

- 3 It is advised that TE and RE are set in the same instruction in order to minimize the setup and the hold time of the receiver.
- 4 The USART supports master mode only: it cannot receive or send data related to an input clock (SCLK is always an output).

Figure 227. USART example of synchronous transmission

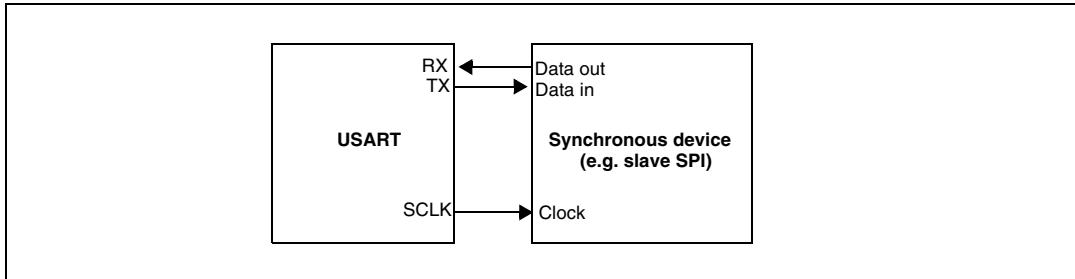


Figure 228. USART data clock timing diagram (M=0)

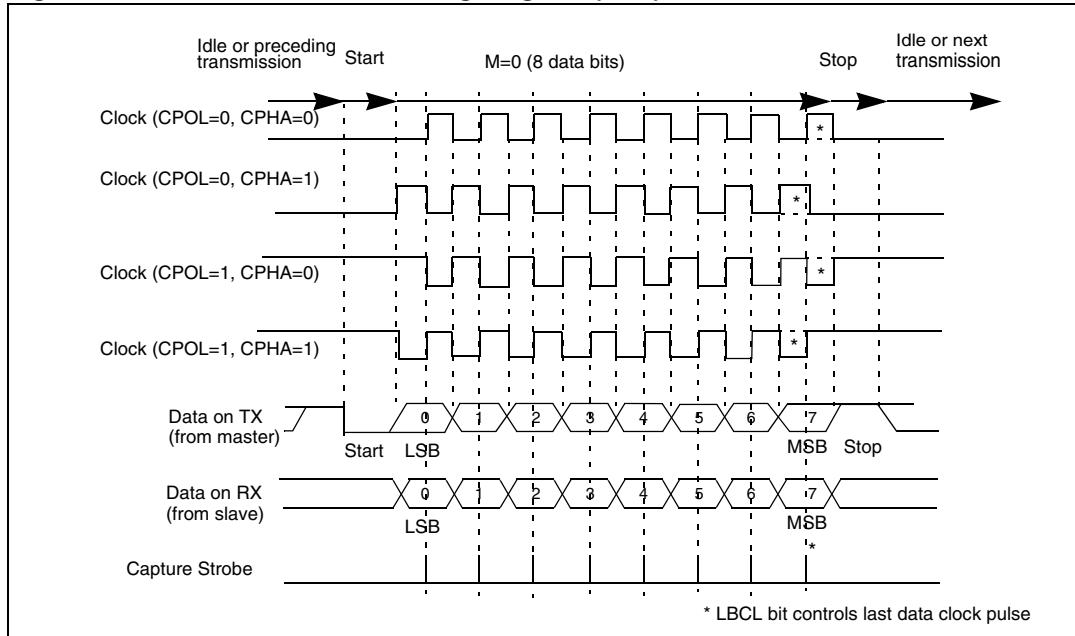
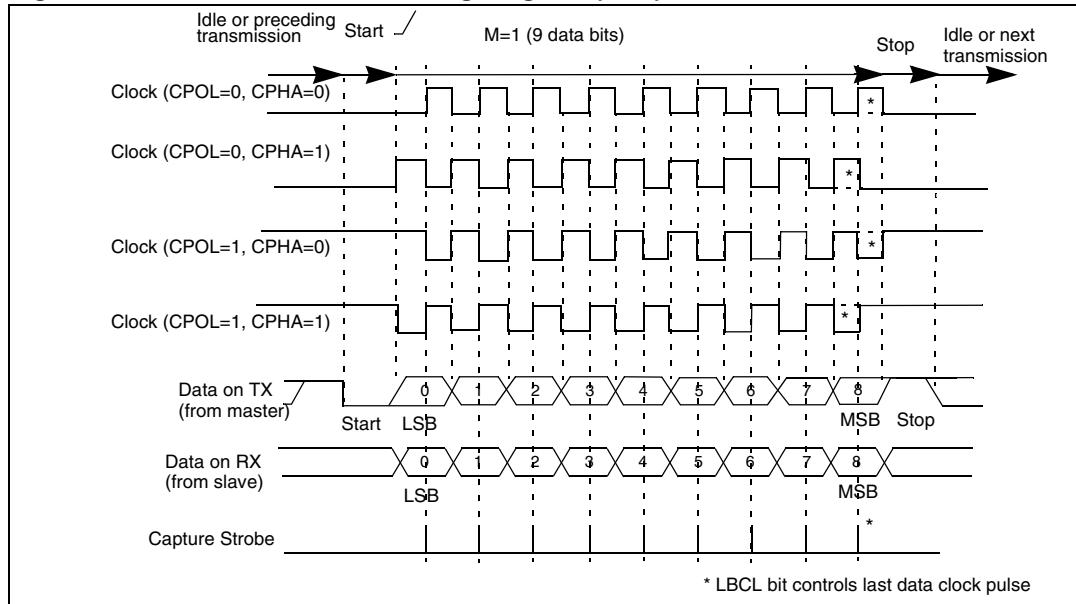
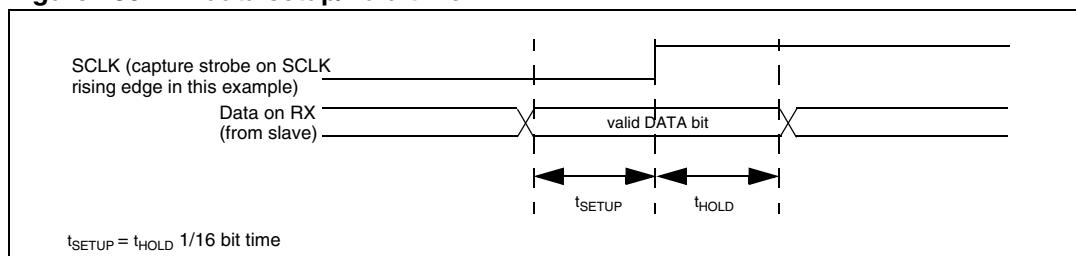


Figure 229. USART data clock timing diagram (M=1)**Figure 230. RX data setup/hold time**

Note:

The function of SCLK is different in Smartcard mode. Refer to the Smartcard mode chapter for more details.

24.3.10 Single-wire half-duplex communication

The single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit 'HALF DUPLEX SEL' (HDSEL in USART_CR3).

As soon as HDSEL is written to 1:

- the TX and RX lines are internally connected
- the RX pin is no longer used
- the TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as floating input (or output high open-drain) when not driven by the USART.

Apart from this, the communications are similar to what is done in normal USART mode. The conflicts on the line must be managed by the software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continue to occur as soon as a data is written in the data register while the TE bit is set.

24.3.11 Smartcard

The Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

Moreover, the CLKEN bit may be set in order to provide a clock to the smartcard.

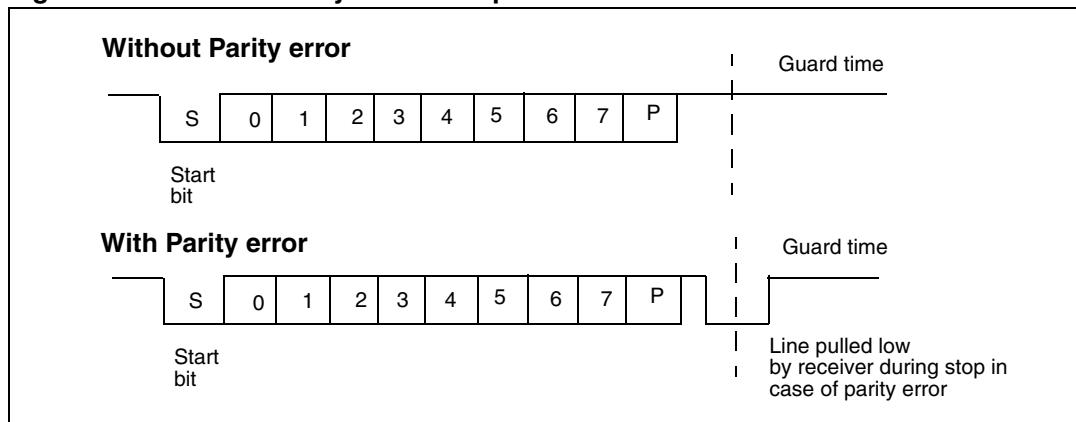
The Smartcard interface is designed to support asynchronous protocol Smartcards as defined in the ISO 7816-3 standard. The USART should be configured as:

- 8 bits plus parity: where M=1 and PCE=1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving: where STOP=11 in the USART_CR2 register.

Note: It is also possible to choose 0.5 stop bit for receiving but it is recommended to use 1.5 stop bits for both transmitting and receiving to avoid switching between the two configurations.

[Figure 231](#) shows examples of what can be seen on the data line with and without parity error.

Figure 231. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that the smartcard also drives into. To do so, SW_RX must be connected on the same I/O than TX at product level. The Transmitter output enable TX_EN is asserted during the transmission of the start bit and the data byte, and is deasserted during the stop bit (weak pull up), so that the receive can drive the line in case of a parity error. If TX_EN is not used, TX is driven at high level during the stop bit. Thus the receiver can drive the line as long as TX is configured in open-drain.

Smartcard is a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register will start shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.

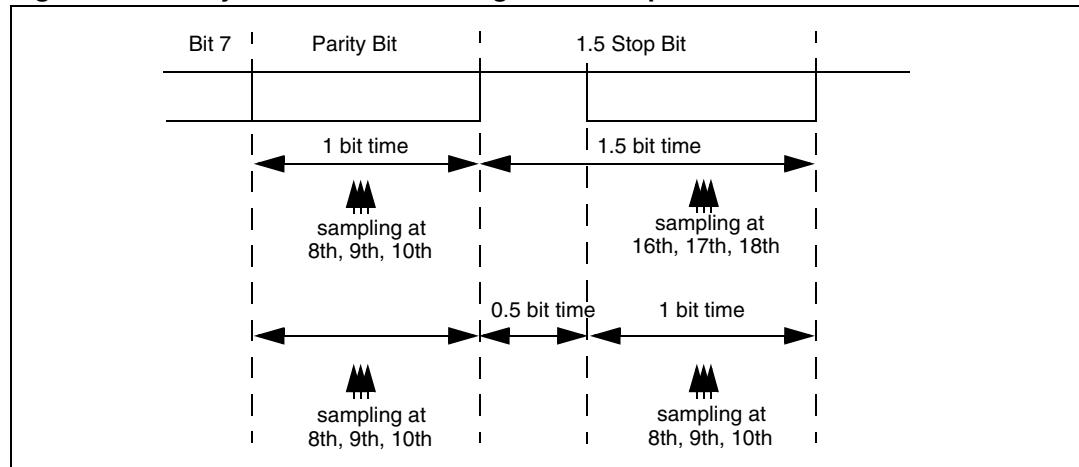
- If a parity error is detected during reception of a frame programmed with a 0.5 or 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to USART has not been correctly received. This NACK signal (pulling transmit line low for 1 baud clock) will cause a framing error on the transmitter side (configured with 1.5 stop bits). The application can handle re-sending of data according to the protocol. A parity error is ‘NACK’ed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted.
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high.
- The de-assertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK will not be detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver will not detect the NACK as a start bit.

Note:

- 1 *A break character is not significant in Smartcard mode. A 0x00 data with a framing error will be treated as data and not as a break.*
- 2 *No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.*

[Figure 232](#) details how the NACK signal is sampled by the USART. In this example the USART is transmitting a data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 232. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the SCLK output. In smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the prescaler register USART_GTPR. SCLK frequency can be programmed from $f_{CK}/2$ to $f_{CK}/62$, where f_{CK} is the peripheral input clock.

24.3.12 IrDA SIR ENDEC block

The IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see [Figure 233](#)).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to USART. The decoder input is normally HIGH (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (i.e. the USART is sending data to the IrDA encoder), any data on the IrDA receive line will be ignored by the IrDA decoder and if the Receiver is busy (USART is receiving decoded data from the USART), data on the TX from the USART to IrDA will not be encoded by IrDA. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.
- A '0 is transmitted as a high pulse and a '1 is transmitted as a '0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see [Figure 234](#)).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 us. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the IrDA low-power Baud Register, USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the STOP bits in the USART_CR2 register must be configured to "1 stop bit".

IrDA low-power mode

Transmitter:

In low-power mode the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally this value is 1.8432 MHz ($1.42\text{ MHz} < \text{PSC} < 2.12\text{ MHz}$). A low-power mode programmable divisor divides the system clock to achieve this value.

Receiver:

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than $1/\text{PSC}$. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in USART_GTPR).

- Note:
- 1 *A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.*
 - 2 *The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).*

Figure 233. IrDA SIR ENDEC- block diagram

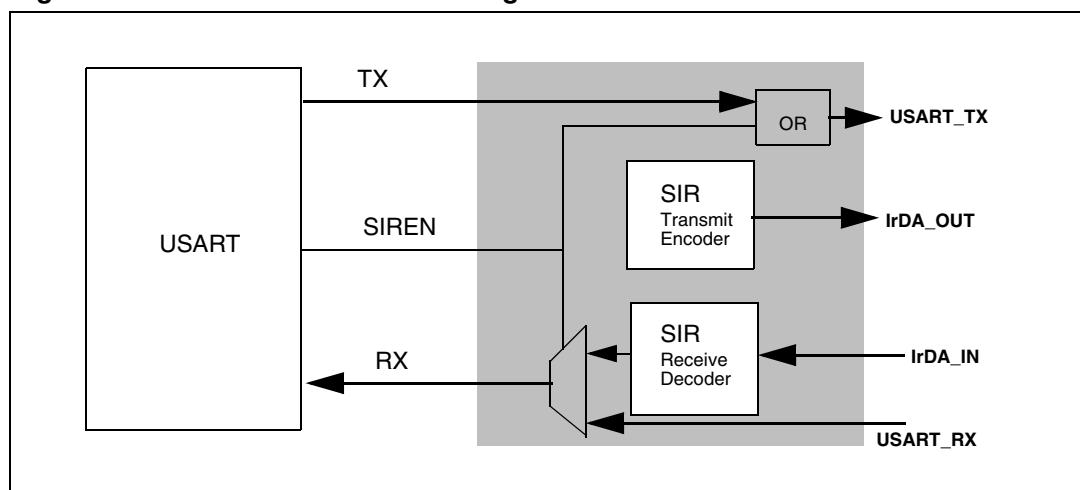
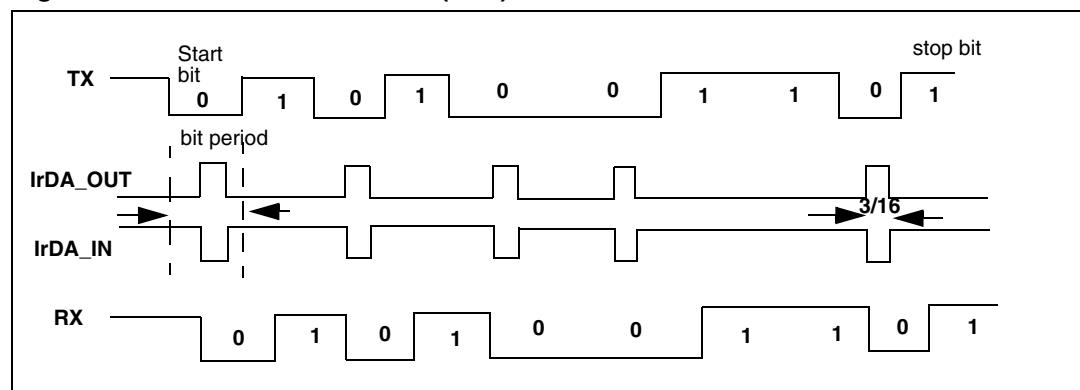


Figure 234. IrDA data modulation (3/16) -Normal mode



24.3.13 Continuous communication using DMA

The USART is capable of continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: You should refer to product specs for availability of the DMA controller. If DMA is not available in the product, you should use the USART as explained in [Section 24.3.2](#) or [24.3.3](#). In the USART_SR register, you can clear the TXE/ RXNE flags to achieve continuous communication.

Transmission using DMA

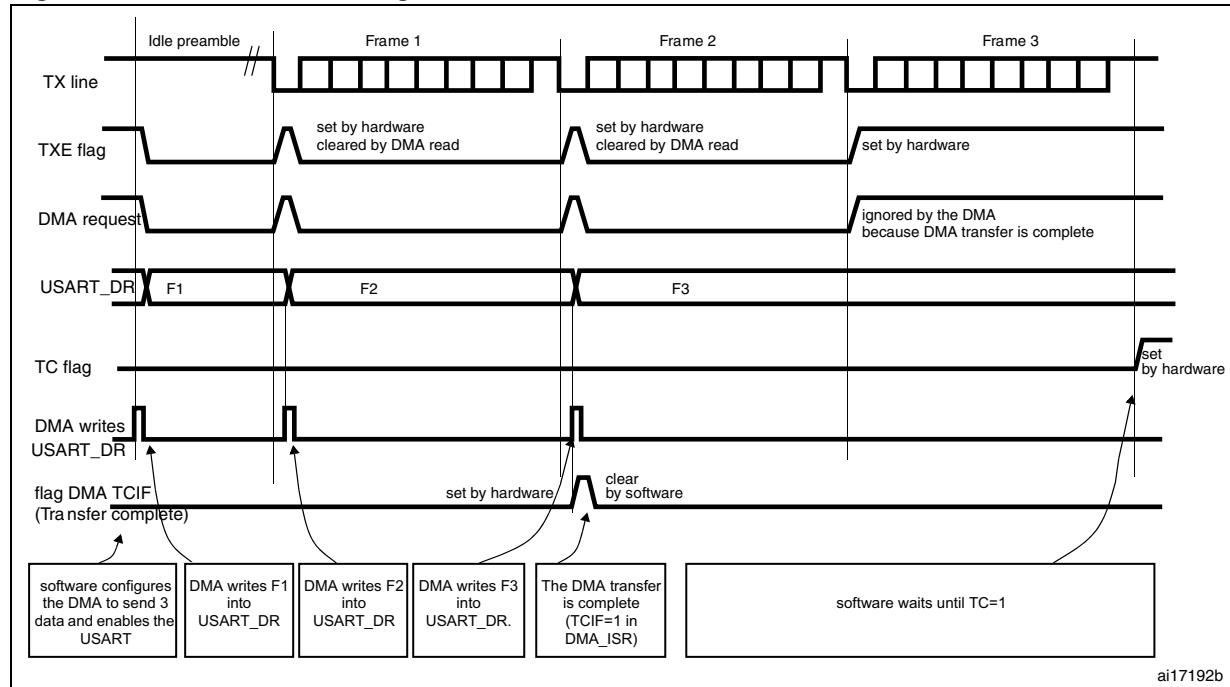
DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to the DMA specification) to the USART_DR register whenever the TXE bit is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_DR register address in the DMA control register to configure it as the destination of the transfer. The data will be moved to this address from memory after each TXE event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data will be loaded into the USART_DR register from this memory area after each TXE event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC bit in the SR register by writing 0 to it.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or entering the Stop mode. The software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the last frame's end of transmission.

Figure 235. Transmission using DMA



ai17192b

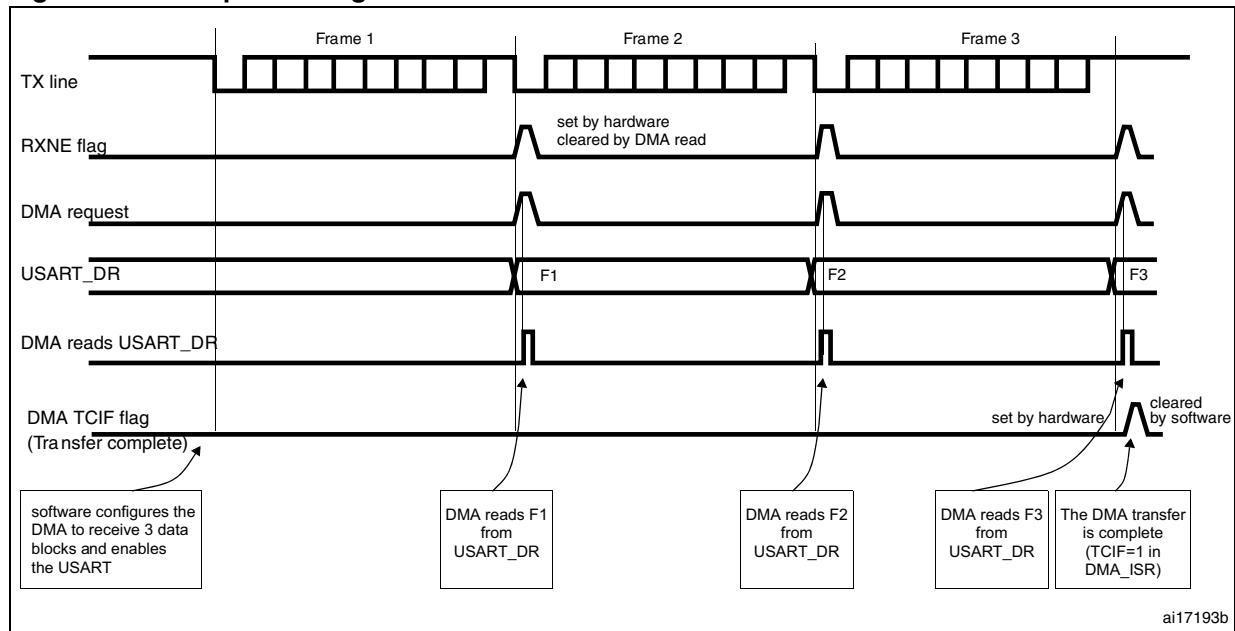
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data is loaded from the USART_DR register to a SRAM area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_DR register address in the DMA control register to configure it as the source of the transfer. The data will be moved from this address to the memory after each RXNE event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data will be loaded from USART_DR to this memory area after each RXNE event.
3. Configure the total number of bytes to be transferred in the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector. The DMAR bit should be cleared by software in the USART_CR3 register during the interrupt subroutine.

Note: If DMA is used for reception, do not enable the RXNEIE bit.

Figure 236. Reception using DMA

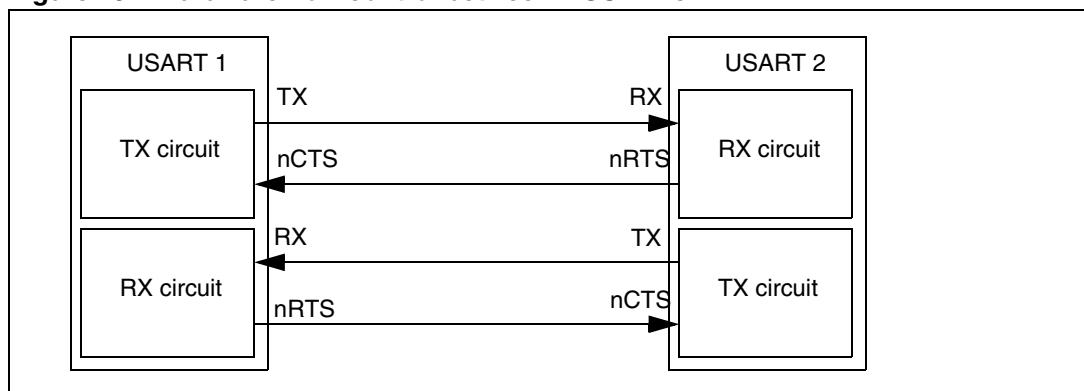
ai17193b

Error flagging and interrupt generation in multibuffer communication

In case of multibuffer communication if any error occurs during the transaction the error flag will be asserted after the current byte. An interrupt will be generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE in case of single byte reception, there will be separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which if set will issue an interrupt after the current byte with either of these errors.

24.3.14 Hardware flow control

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The [Figure 237](#) shows how to connect 2 devices in this mode:

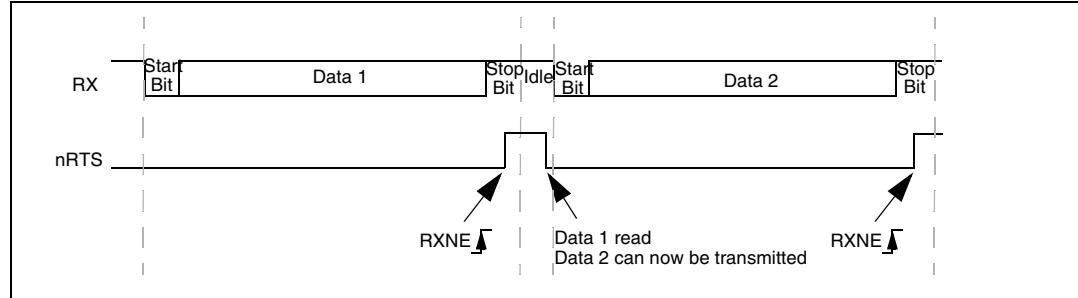
Figure 237. Hardware flow control between 2 USARTs

RTS and CTS flow control can be enabled independently by writing respectively RTSE and CTSE bits to 1 (in the USART_CR3 register).

RTS flow control

If the RTS flow control is enabled ($RTSE=1$), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. [Figure 238](#) shows an example of communication with RTS flow control enabled.

Figure 238. RTS flow control

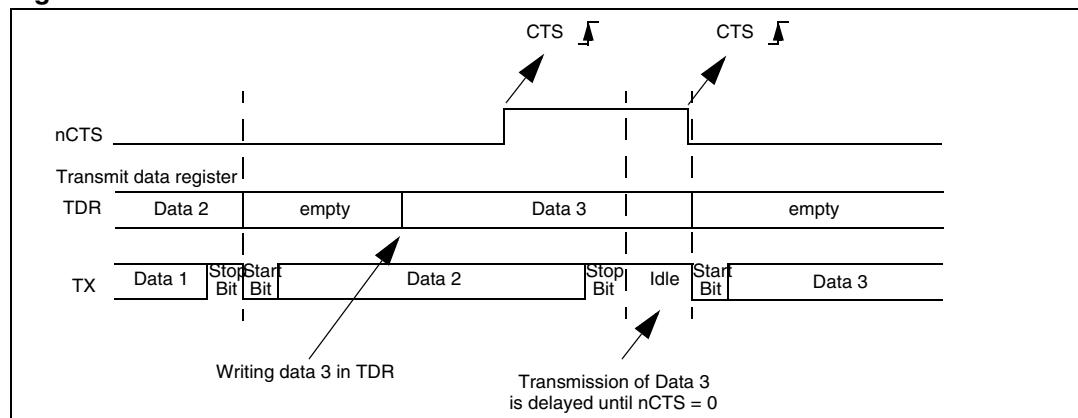


CTS flow control

If the CTS flow control is enabled ($CTSE=1$), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that a data is to be transmitted, in other words, if $TXE=0$), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When $CTSE=1$, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. The figure below shows an example of communication with CTS flow control enabled.

Figure 239. CTS flow control



Note:

Special behavior of break frames: when the CTS flow is enabled, the transmitter does not check the nCTS input state to send a break.

24.4 USART interrupts

Table 96. USART interrupt requests

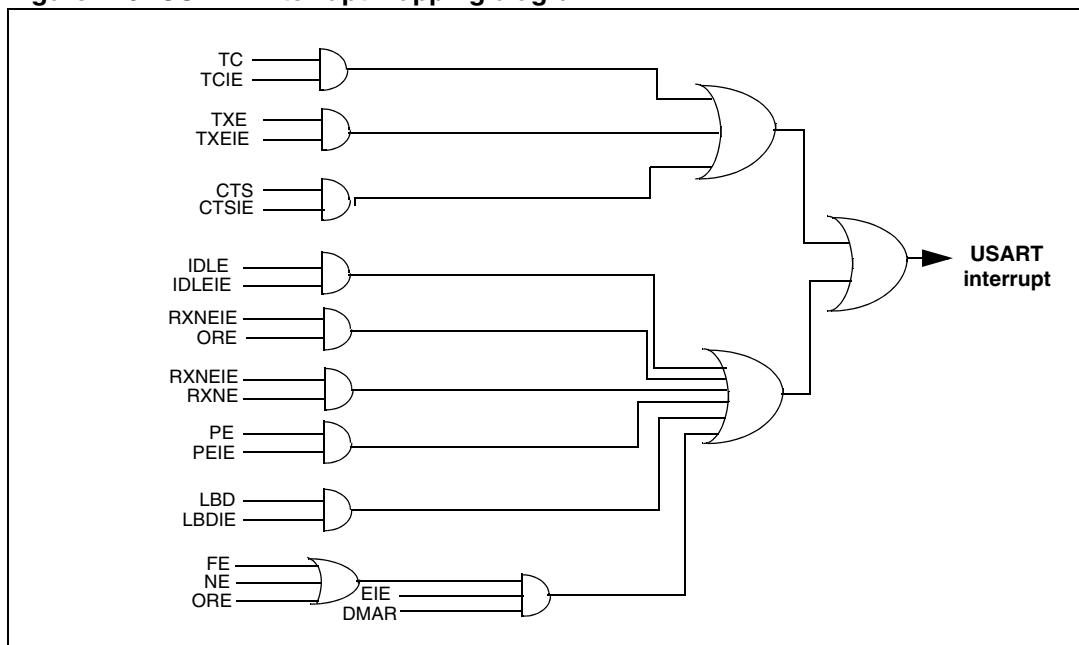
Interrupt event	Event flag	Enable control bit
Transmit Data Register Empty	TXE	TXEIE
CTS flag	CTS	CTSIE
Transmission Complete	TC	TCIE
Received Data Ready to be Read	RXNE	RXNEIE
Overrun Error Detected	ORE	
Idle Line Detected	IDLE	IDLEIE
Parity Error	PE	PEIE
Break Flag	LBD	LBDIE
Noise Flag, Overrun error and Framing Error in multibuffer communication	NF or ORE or FE	EIE

The USART interrupt events are connected to the same interrupt vector (see [Figure 240](#)).

- During transmission: Transmission Complete, Clear to Send or Transmit Data Register empty interrupt.
- While receiving: Idle Line detection, Overrun error, Receive Data register not empty, Parity error, LIN break detection, Noise Flag (only in multi buffer communication) and Framing Error (only in multi buffer communication).

These events generate an interrupt if the corresponding Enable Control Bit is set.

Figure 240. USART interrupt mapping diagram



24.5 USART mode configuration

Table 97. USART mode configuration⁽¹⁾

USART modes	USART1	USART2	USART3	UART4	UART5	USART6
Asynchronous mode	X	X	X	X	X	X
Hardware flow control	X	X	X	NA	NA	X
Multibuffer communication (DMA)	X	X	X	X	X	X
Multiprocessor communication	X	X	X	X	X	X
Synchronous	X	X	X	NA	NA	X
Smartcard	X	X	X	NA	NA	X
Half-duplex (single-wire mode)	X	X	X	X	X	X
IrDA	X	X	X	X	X	X
LIN	X	X	X	X	X	X

1. X = supported; NA = not applicable.

24.6 USART registers

Refer to [Section 1.1 on page 46](#) for a list of abbreviations used in register descriptions.

24.6.1 Status register (USART_SR)

Address offset: 0x00

Reset value: 0x00C0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE	
					rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r	

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

Bit 9 CTS: CTS flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software (by writing it to 0). An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the nCTS status line
1: A change occurred on the nCTS status line

Note: This bit is not available for UART4 & UART5.

Bit 8 LBD: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software (by writing it to 0). An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected
1: LIN break detected

Note: An interrupt is generated when LBD=1 if LBDIE=1

Bit 7 TXE: Transmit data register empty

This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register. It is cleared by a write to the USART_DR register.

- 0: Data is not transferred to the shift register
- 1: Data is transferred to the shift register

Note: This bit is used during single buffer transmission.

Bit 6 TC: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE=1 in the USART_CR1 register. It is cleared by a software sequence (a read from the USART_SR register followed by a write to the USART_DR register). The TC bit can also be cleared by writing a '0' to it. This clearing sequence is recommended only for multibuffer communication.

- 0: Transmission is not complete
- 1: Transmission is complete

Bit 5 RXNE: Read data register not empty

This bit is set by hardware when the content of the RDR shift register has been transferred to the USART_DR register. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. This clearing sequence is recommended only for multibuffer communication.

- 0: Data is not received
- 1: Received data is ready to be read.

Bit 4 IDLE: IDLE line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if the IDLEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

- 0: No Idle Line is detected
- 1: Idle Line is detected

Note: The IDLE bit will not be set again until the RXNE bit has been set itself (i.e. a new idle line occurs).

Bit 3 ORE: Overrun error

This bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register while RXNE=1. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

- 0: No Overrun error
- 1: Overrun error is detected

Note: When this bit is set, the RDR register content will not be lost but the shift register will be overwritten. An interrupt is generated on ORE flag in case of Multi Buffer communication if the EIE bit is set.

Bit 2 **NF**: Noise detected flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

- 0: No noise is detected
- 1: Noise is detected

Note: This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. If the word currently being transferred causes both noise detection and overrun error, it will be transferred and only the ORE bit will be 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 24.3.5: USART receiver tolerance to clock deviation on page 628](#)).

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

- 0: No Framing error is detected
- 1: Framing error or break character is detected

Note: This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. If the word currently being transferred causes both frame error and overrun error, it will be transferred and only the ORE bit will be set.

An interrupt is generated on FE flag in case of Multi Buffer communication if the EIE bit is set.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by a software sequence (a read from the status register followed by a read or write access to the USART_DR data register). The software must wait for the RXNE flag to be set before clearing the PE bit.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

- 0: No parity error
- 1: Parity error

24.6.2 Data register (USART_DR)

Address offset: 0x04

Reset value: 0xFFFF XXXX

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

Bits 8:0 **DR[8:0]**: Data value

Contains the Received or Transmitted data character, depending on whether it is read from or written to.

The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR)

The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 1).

The RDR register provides the parallel interface between the input shift register and the internal bus.

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

24.6.3 Baud rate register (USART_BRR)

Note: The baud counters stop counting if the TE or RE bits are disabled respectively.

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bits 15:4 **DIV_Mantissa[11:0]**: mantissa of USARTDIV

These 12 bits define the mantissa of the USART Divider (USARTDIV)

Bits 3:0 **DIV_Fraction[3:0]**: fraction of USARTDIV

These 4 bits define the fraction of the USART Divider (USARTDIV). When OVER8=1, the DIV_Fraction3 bit is not considered and must be kept cleared.

24.6.4 Control register 1 (USART_CR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw	Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 15 **OVER8**: Oversampling mode

0: oversampling by 16

1: oversampling by 8

Note: Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes: when SCEN=1,IREN=1 or LINEN=1 then OVER8 is forced to '0' by hardware.

Bit 14 Reserved, must be kept at reset value

Bit 13 **UE**: USART enable

When this bit is cleared the USART prescalers and outputs are stopped and the end of the current

byte transfer in order to reduce power consumption. This bit is set and cleared by software.

0: USART prescaler and outputs disabled

1: USART enabled

Bit 12 **M**: Word length

This bit determines the word length. It is set or cleared by software.

0: 1 Start bit, 8 Data bits, n Stop bit

1: 1 Start bit, 9 Data bits, n Stop bit

Note: The M bit must not be modified during a data transfer (both transmission and reception)

Bit 11 WAKE: Wakeup method

This bit determines the USART wakeup method, it is set or cleared by software.

0: Idle Line

1: Address Mark

Bit 10 PCE: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

Bit 9 PS: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.

0: Even parity

1: Odd parity

Bit 8 PEIE: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever PE=1 in the USART_SR register

Bit 7 TXEIE: TXE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever TXE=1 in the USART_SR register

Bit 6 TCIE: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever TC=1 in the USART_SR register

Bit 5 RXNEIE: RXNE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever ORE=1 or RXNE=1 in the USART_SR register

Bit 4 IDLEIE: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever IDLE=1 in the USART_SR register

Bit 3 TE: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: 1: During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word, except in smartcard mode.

2: When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 RE: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 RWU: Receiver wakeup

This bit determines if the USART is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wakeup sequence is recognized.

0: Receiver in active mode

1: Receiver in mute mode

Note: 1: Before selecting Mute mode (by setting the RWU bit) the USART must first receive a data byte, otherwise it cannot function in Mute mode with wakeup by Idle line detection.

2: In Address Mark Detection wakeup configuration (WAKE bit=1) the RWU bit cannot be modified by software while the RXNE bit is set.

Bit 0 SBK: Send break

This bit set is used to send break characters. It can be set and cleared by software. It should be set by software, and will be reset by hardware during the stop bit of break.

0: No break character is transmitted

1: Break character will be transmitted

24.6.5 Control register 2 (USART_CR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

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

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 SBK bit in the USART_CR1 register, and to detect LIN Sync breaks.

Bits 13:12 **STOP**: STOP bits

These bits are used for programming the stop bits.

00: 1 Stop bit

01: 0.5 Stop bit

10: 2 Stop bits

11: 1.5 Stop bit

Note: The 0.5 Stop bit and 1.5 Stop bit are not available for UART4 & UART5.

Bit 11 **CLKEN**: Clock enable

This bit allows the user to enable the SCLK pin.

0: SCLK pin disabled

1: SCLK pin enabled

This bit is not available for UART4 & UART5.

Bit 10 **CPOL**: Clock polarity

This bit allows the user to select the polarity of the clock output on the 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 is not available for UART4 & UART5.

Bit 9 **CPHA**: Clock phase

This bit allows the user to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see figures 228 to 229)

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 is not available for UART4 & UART5.

Bit 8 **LBCL**: Last bit clock pulse

This bit allows the user to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the 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

Note: 1: The last bit is the 8th or 9th data bit transmitted depending on the 8 or 9 bit format selected by the M bit in the USART_CR1 register.

2: This bit is not available for UART4 & UART5.

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 LBD=1 in the USART_SR register

Bit 5 **LBDL**: lin break detection length

This bit is for selection between 11 bit or 10 bit break detection.

- 0: 10-bit break detection
- 1: 11-bit break detection

Bit 4 Reserved, must be kept at reset value

Bits 3:0 **ADD[3:0]**: Address of the USART node

This bit-field gives the address of the USART node.

This is used in multiprocessor communication during mute mode, for wake up with address mark detection.

Note: These 3 bits (CPOL, CPHA, LBCL) should not be written while the transmitter is enabled.

24.6.6 Control register 3 (USART_CR3)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

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

Bit 10 **CTSIE**: CTS interrupt enable

- 0: Interrupt is inhibited

- 1: An interrupt is generated whenever CTS=1 in the USART_SR register

Note: This bit is not available for UART4 & UART5.

Bit 9 CTSE: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while a data is being transmitted, then the transmission is completed before stopping. If a data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.

Note: This bit is not available for UART4 & UART5.

Bit 8 RTSE: RTS enable

0: RTS hardware flow control disabled

1: RTS interrupt enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (tied to 0) when a data can be received.

Note: This bit is not available for UART4 & UART5.

Bit 7 DMAT: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission.

0: DMA mode is disabled for transmission.

Note: This bit is not available for UART5.

Bit 6 DMAR: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Note: This bit is not available for UART5.

Bit 5 SCEN: Smartcard mode enable

This bit is used for enabling Smartcard mode.

0: Smartcard Mode disabled

1: Smartcard Mode enabled

Note: This bit is not available for UART4 & UART5.

Bit 4 NACK: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

Note: This bit is not available for UART4 & UART5.

Bit 3 HDSEL: Half-duplex selection

Selection of Single-wire Half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

Bit 2 IRLP: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

Bit 1 IREN: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

Bit 0 EIE: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NF=1 in the USART_SR register) in case of Multi Buffer Communication (DMAR=1 in the USART_CR3 register).

0: Interrupt is inhibited

1: An interrupt is generated whenever DMAR=1 in the USART_CR3 register and FE=1 or ORE=1 or NF=1 in the USART_SR register.

24.6.7 Guard time and prescaler register (USART_GTPR)

Address offset: 0x18

Reset value: 0x0000 0000

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

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

Bits 15:8 **GT[7:0]**: Guard time value

This bit-field gives the Guard time value in terms of number of baud clocks.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

Note: This bit is not available for UART4 & UART5.

Bits 7:0 **PSC[7:0]**: Prescaler value

– In IrDA Low-power mode:

PSC[7:0] = IrDA Low-Power Baud Rate

Used for programming the prescaler for dividing the system clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

– In normal IrDA mode: PSC must be set to 00000001.

– In smartcard mode:

PSC[4:0]: Prescaler value

Used for programming the prescaler for dividing the system clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

Note: 1: Bits [7:5] have no effect if Smartcard mode is used.

2: This bit is not available for UART4 & UART5.

24.6.8 USART register map

The table below gives the USART register map and reset values.

Table 98. USART register map and reset values

Refer to [Table 1](#) on page 50 for the register boundary addresses.

25 Serial peripheral interface (SPI)

This section applies to the whole STM32F40x and STM32F41x family, unless otherwise specified.

25.1 SPI introduction

The SPI interface gives the flexibility to get either the SPI protocol or the I²S audio protocol. By default, it is the SPI function that is selected. It is possible to switch the interface from SPI to I²S by software. The serial peripheral interface (SPI) allows half/ full-duplex, synchronous, serial communication with external devices. The interface can be configured as the master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

It may be used for a variety of purposes, including simplex synchronous transfers on two lines with a possible bidirectional data line or reliable communication using CRC checking.

The I²S is also a synchronous serial communication interface. It can address four different audio standards including the I²S Phillips standard, the MSB- and LSB-justified standards, and the PCM standard. It can operate as a slave or a master device in full-duplex mode (using 4 pins) or in half-duplex mode (using 6 pins). Master clock can be provided by the interface to an external slave component when the I²S is configured as the communication master.

Warning: Since some SPI1 and SPI3/I2S3 pins may be mapped onto some pins used by the JTAG interface (SPI1_NSS onto JTDI, SPI3_NSS/I2S3_WS onto JTDO and SPI3_SCK/I2S3_CK onto JTDO), you may either:

- map SPI/I2S onto other pins
- disable the JTAG and use the SWD interface prior to configuring the pins listed as SPI I/Os (when debugging the application) or
- disable both JTAG/SWD interfaces (for standalone applications).

For more information on the configuration of the JTAG/SWD interface pins, please refer to [Section 6.3.2: I/O pin multiplexer and mapping](#).

25.2 SPI and I²S main features

25.2.1 SPI features

- Full-duplex synchronous transfers on three lines
- Simplex synchronous transfers on two lines with or without a bidirectional data line
- 8- or 16-bit transfer frame format selection
- Master or slave operation
- Multimaster mode capability
- 8 master mode baud rate prescalers ($f_{PCLK}/2$ max.)
- Slave mode frequency ($f_{PCLK}/2$ max)
- Faster communication for both master and slave
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI TI mode
- Hardware CRC feature for reliable communication:
 - CRC value can be transmitted as last byte in Tx mode
 - Automatic CRC error checking for last received byte
- Master mode fault, overrun and CRC error flags with interrupt capability
- 1-byte transmission and reception buffer with DMA capability: Tx and Rx requests

25.2.2 I²S features

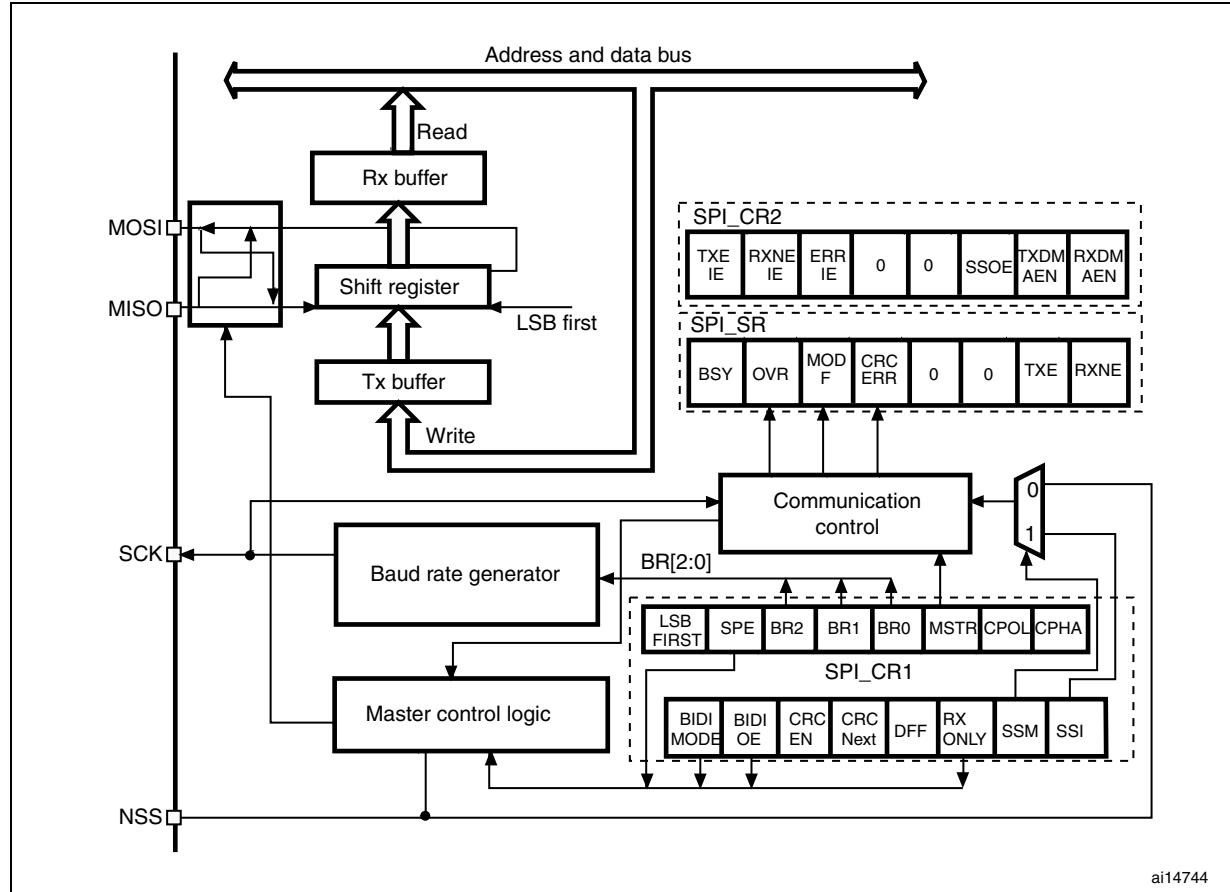
- Full duplex communication
- Simplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies (from 8 kHz to 192 kHz)
- Data format may be 16-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit 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 and Overrun flag in reception mode (master and slave)
- 16-bit register for transmission and reception with one data register for both channel sides
- Supported I²S protocols:
 - I²S Phillips 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 may be output to drive an external audio component. Ratio is fixed at $256 \times F_S$ (where F_S is the audio sampling frequency)
- Both I²S (I2S2 and I2S3) have a dedicated PLL (PLLI2S) to generate an even more accurate clock.
- I²S (I2S2 and I2S3) clock can be derived from an external clock mapped on the I2S_CKIN pin.

25.3 SPI functional description

25.3.1 General description

The block diagram of the SPI is shown in [Figure 241](#).

Figure 241. SPI block diagram



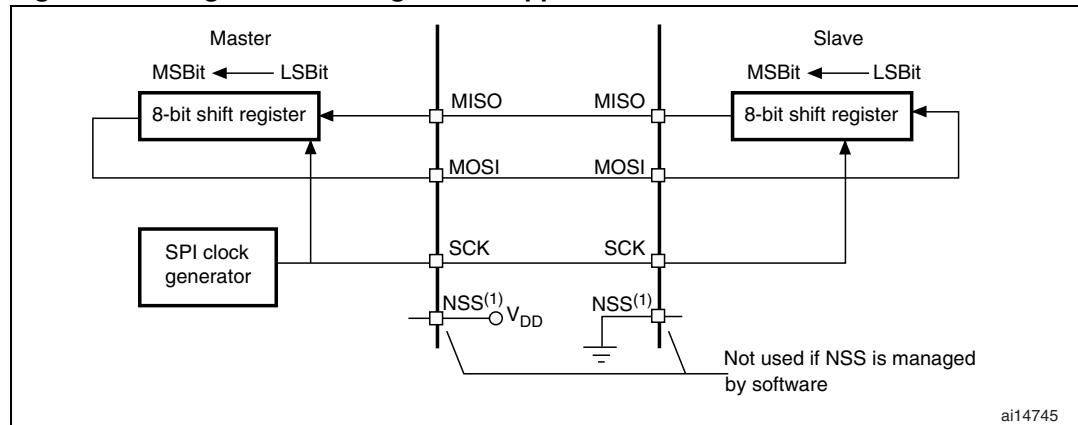
Usually, the SPI is connected to external devices through 4 pins:

- MISO: Master In / Slave Out data. This pin can be used to transmit data in slave mode and receive data in master mode.
- MOSI: Master Out / Slave In data. This pin can be used to transmit data in master mode and receive data in slave mode.
- SCK: Serial Clock output for SPI masters and input for SPI slaves.
- NSS: Slave select. This is an optional pin to select a slave device. This pin acts as a ‘chip select’ to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave NSS inputs can be driven by standard IO ports on the master device. The NSS pin may also be used as an output if enabled (SSOE bit) and driven low if the SPI is in master configuration. In this manner, all NSS pins from devices connected to the Master NSS pin see a low level and become slaves when they are configured in NSS hardware mode. When configured in master mode with NSS configured as an input (MSTR=1 and SSOE=0) and if NSS is pulled low, the SPI

enters the master mode fault state: the MSTR bit is automatically cleared and the device is configured in slave mode (refer to [Section 25.3.10: Error flags on page 682](#)).

A basic example of interconnections between a single master and a single slave is illustrated in [Figure 242](#).

Figure 242. Single master/ single slave application



ai14745

1. Here, the NSS pin is configured as an input.

The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

The communication is always initiated by the master. When the master device transmits data to a slave device via the MOSI pin, the slave device responds via the MISO pin. This implies full-duplex communication with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

Slave select (NSS) pin management

Hardware or software slave select management can be set using the SSM bit in the SPI_CR1 register.

- Software NSS management (SSM = 1)

The slave select information is driven internally by the value of the SSI bit in the SPI_CR1 register. The external NSS pin remains free for other application uses.

- Hardware NSS management (SSM = 0)

Two configurations are possible depending on the NSS output configuration (SSOE bit in register SPI_CR1).

- NSS output enabled (SSM = 0, SSOE = 1)

This configuration is used only when the device operates in master mode. The NSS signal is driven low when the master starts the communication and is kept low until the SPI is disabled.

- NSS output disabled (SSM = 0, SSOE = 0)

This configuration allows multimaster capability for devices operating in master mode. For devices set as slave, the NSS pin acts as a classical NSS input: the slave is selected when NSS is low and deselected when NSS high.

Clock phase and clock polarity

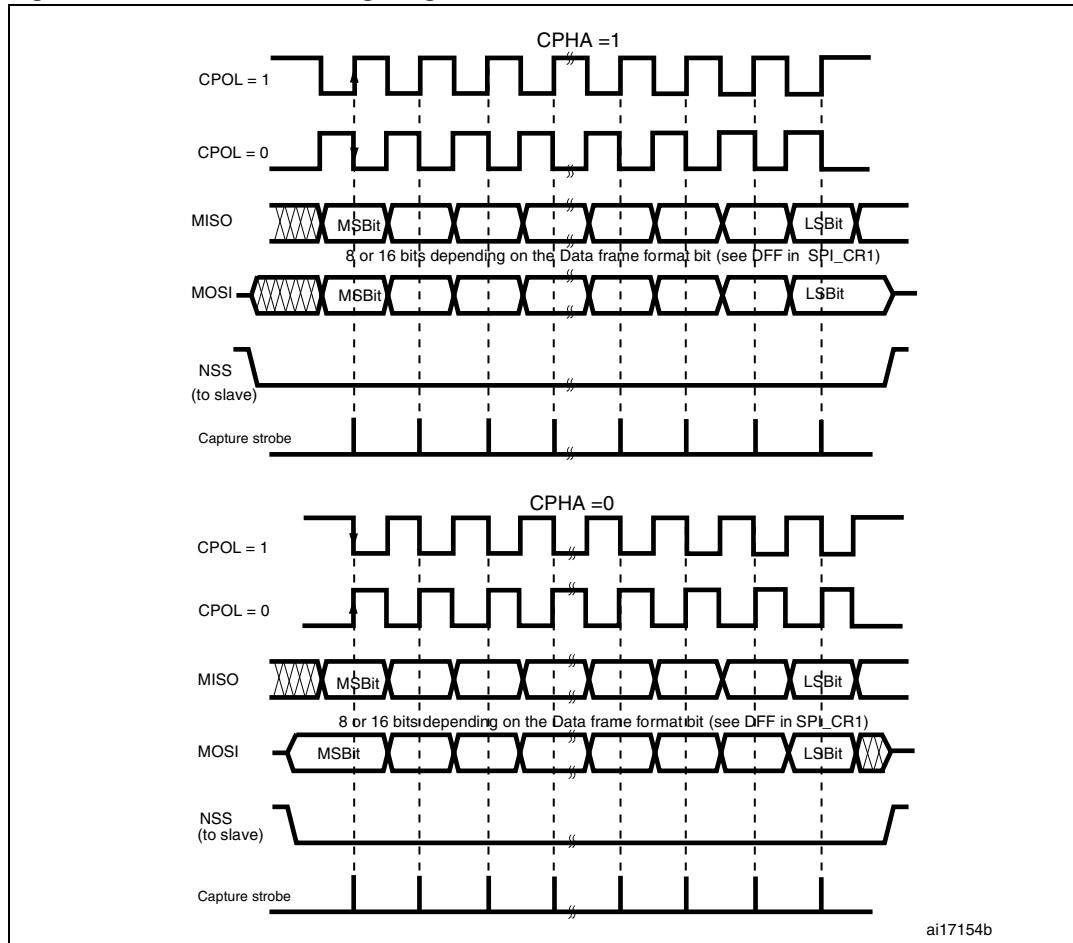
Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI_CR1 register. The CPOL (clock polarity) bit controls the steady state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA (clock phase) bit is set, the second edge on the SCK pin (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set) is the MSBit capture strobe. Data are latched on the occurrence of the second clock transition. If the CPHA bit is reset, the first edge on the SCK pin (falling edge if CPOL bit is set, rising edge if CPOL bit is reset) is the MSBit capture strobe. Data are latched on the occurrence of the first clock transition.

The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 243, shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin, the MOSI pin are directly connected between the master and the slave device.

- Note:*
- 1 *Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.*
 - 2 *Master and slave must be programmed with the same timing mode.*
 - 3 *The idle state of SCK must correspond to the polarity selected in the SPI_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).*
 - 4 *The Data Frame Format (8- or 16-bit) is selected through the DFF bit in SPI_CR1 register, and determines the data length during transmission/reception.*

Figure 243. Data clock timing diagram

1. These timings are shown with the LSBFIRST bit reset in the SPI_CR1 register.

Data frame format

Data can be shifted out either MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 Register.

Each data frame is 8 or 16 bits long depending on the size of the data programmed using the DFF bit in the SPI_CR1 register. The selected data frame format is applicable for transmission and/or reception.

25.3.2 Configuring the SPI in slave mode

In the slave configuration, the serial clock is received on the SCK pin from the master device. The value set in the BR[2:0] bits in the SPI_CR1 register, does not affect the data transfer rate.

Note: *It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication. It is mandatory to have the polarity of the communication clock set to the steady state value before the slave and the master are enabled.*

Follow the procedure below to configure the SPI in slave mode:

Procedure

1. Set the DFF bit to define 8- or 16-bit data frame format
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see [Figure 243](#)). For correct data transfer, the CPOL and CPHA bits must be configured in the same way in the slave device and the master device. This step is not required when the TI mode is selected through the FRF bit of the SPI_CR2 register.
3. The frame format (MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 register) must be the same as the master device. This step is not required when the TI mode is selected.
4. In Hardware mode (refer to [Slave select \(NSS\) pin management on page 662](#)), the NSS pin must be connected to a low level signal during the complete byte transmit sequence. In NSS software mode, set the SSM bit and clear the SSI bit in the SPI_CR1 register. This step is not required when the TI mode is selected.
5. Set the FRF bit in the SPI_CR2 register to select the TI mode protocol for serial communications.
6. Clear the MSTR bit and set the SPE bit (both in the SPI_CR1 register) to assign the pins to alternate functions.

In this configuration the MOSI pin is a data input and the MISO pin is a data output.

Transmit sequence

The data byte is parallel-loaded into the Tx buffer during a write cycle.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin. The remaining bits (the 7 bits in 8-bit data frame format, and the 15 bits in 16-bit data frame format) are loaded into the shift-register. The TXE flag in the SPI_SR register is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXIE bit in the SPI_CR2 register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The Data in shift register is transferred to Rx Buffer and the RXNE flag (SPI_SR register) is set
- An Interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register.

After the last sampling clock edge the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI_DR register is read, the SPI peripheral returns this buffered value.

Clearing of the RXNE bit is performed by reading the SPI_DR register.

SPI TI protocol in slave mode

In slave mode, the SPI interface is compatible with the TI protocol. The FRF bit of the SPI_CR2 register can be used to configure the slave SPI serial communications 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 SPI_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPI_CR1 and SPI_CR2 registers (such as SSM, SSI, SSOE) transparent for the user.

In Slave mode ([Figure 244: TI mode - Slave mode, single transfer](#) and [Figure 245: TI mode - Slave mode, continuous transfer](#)), the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ. Any baud rate can be used thus allowing to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The time for the MISO signal to become HiZ ($t_{release}$) depends on internal resynchronizations and on the baud rate value set in through BR[2:0] of SPI_CR1 register. It is given by the formula:

$$\frac{t_{baud_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud_rate}}{2} + 6 \times t_{pclk}$$

Note: This feature is not available for Motorola SPI communications (FRF bit set to 0).

To detect TI frame errors in Slave transmitter only mode by using the Error interrupt (ERRIE = 1), the SPI must be configured in 2-line unidirectional mode by setting BIDIMODE and BIDIOE to 1 in the SPI_CR1 register. When BIDIMODE is set to 0, OVR is set to 1 because the data register is never read and error interrupt are always generated, while when BIDIMODE is set to 1, data are not received and OVR is never set.

Figure 244. TI mode - Slave mode, single transfer

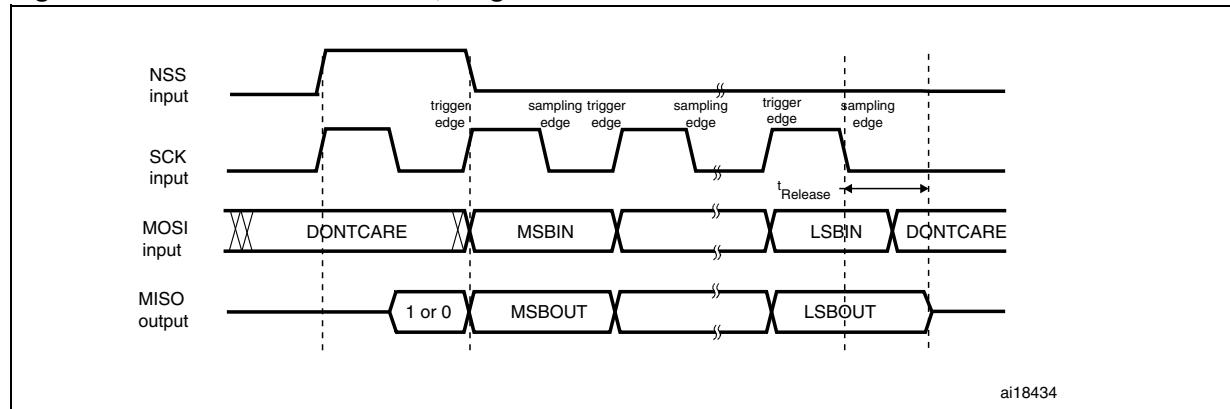
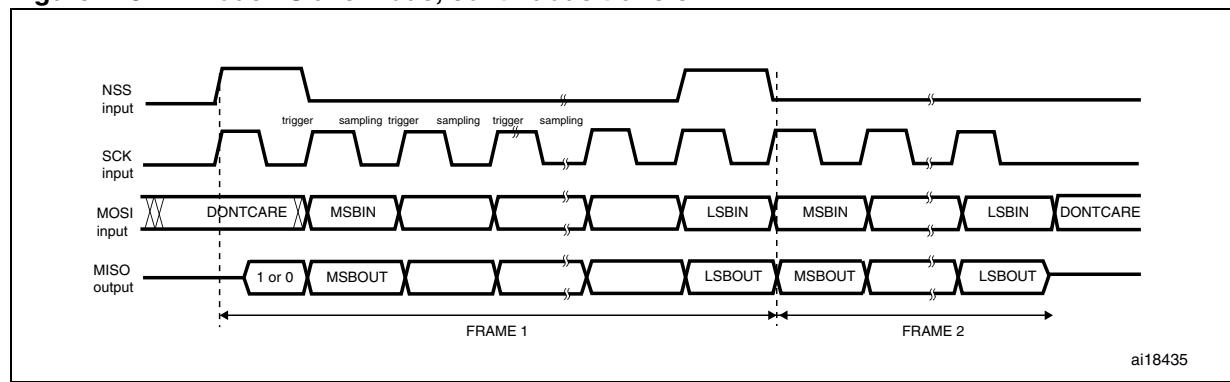


Figure 245. TI mode - Slave mode, continuous transfer



25.3.3 Configuring the SPI in master mode

In the master configuration, the serial clock is generated on the SCK pin.

Procedure

1. Select the BR[2:0] bits to define the serial clock baud rate (see SPI_CR1 register).
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see [Figure 243](#)). This step is not required when the TI mode is selected.
3. Set the DFF bit to define 8- or 16-bit data frame format
4. Configure the LSBFIRST bit in the SPI_CR1 register to define the frame format. This step is not required when the TI mode is selected.
5. If the NSS pin is required in input mode, in hardware mode, connect the NSS pin to a high-level signal during the complete byte transmit sequence. In NSS software mode, set the SSM and SSI bits in the SPI_CR1 register. If the NSS pin is required in output mode, the SSOE bit only should be set. This step is not required when the TI mode is selected.
6. Set the FRF bit in SPI_CR2 to select the TI protocol for serial communications.
7. The MSTR and SPE bits must be set (they remain set only if the NSS pin is connected to a high-level signal).

In this configuration the MOSI pin is a data output and the MISO pin is a data input.

Transmit sequence

The transmit sequence begins when a byte is written in the Tx Buffer.

The data byte is parallel-loaded into the shift register (from the internal bus) during the first bit transmission and then shifted out serially to the MOSI pin MSB first or LSB first depending on the LSBFIRST bit in the SPI_CR1 register. The TXE flag is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The data in the shift register is transferred to the RX Buffer and the RXNE flag is set
- An interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register

At the last sampling clock edge the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI_DR register is read, the SPI peripheral returns this buffered value.

Clearing the RXNE bit is performed by reading the SPI_DR register.

A continuous transmit stream can be maintained if the next data to be transmitted is put in the Tx buffer once the transmission is started. Note that TXE flag should be ‘1’ before any attempt to write the Tx buffer is made.

Note:

When a master is communicating with SPI slaves which need to be de-selected between transmissions, the NSS pin must be configured as GPIO or another GPIO must be used and toggled by software.

SPI TI protocol in master mode

In master mode, the SPI interface is compatible with the TI protocol. The FRF bit of the SPI_CR2 register can be used to configure the master SPI serial communications 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 SPI_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPI_CR1 and SPI_CR2 registers (SSM, SSI, SSOE) transparent for the user.

Figure 246: TI mode - master mode, single transfer and *Figure 247: TI mode - master mode, continuous transfer* show the SPI master communication waveforms when the TI mode is selected in master mode.

Figure 246. TI mode - master mode, single transfer

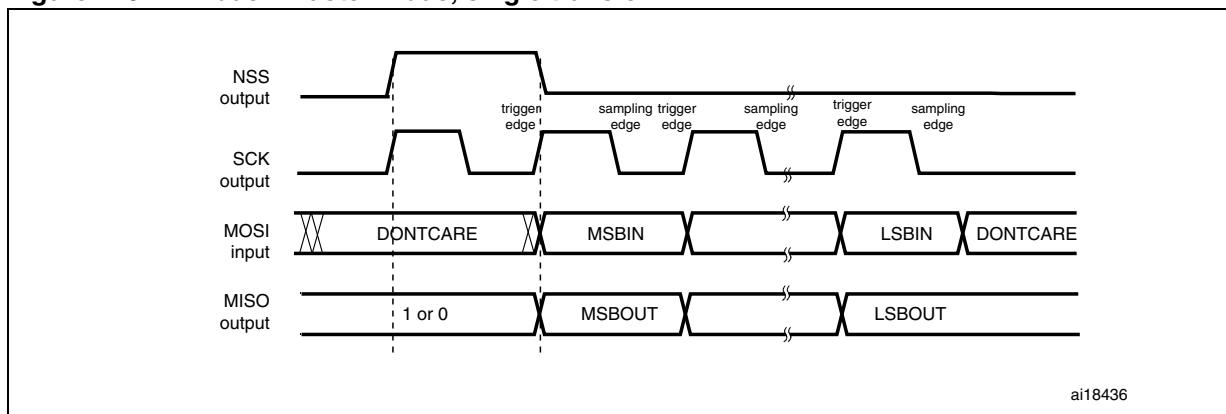
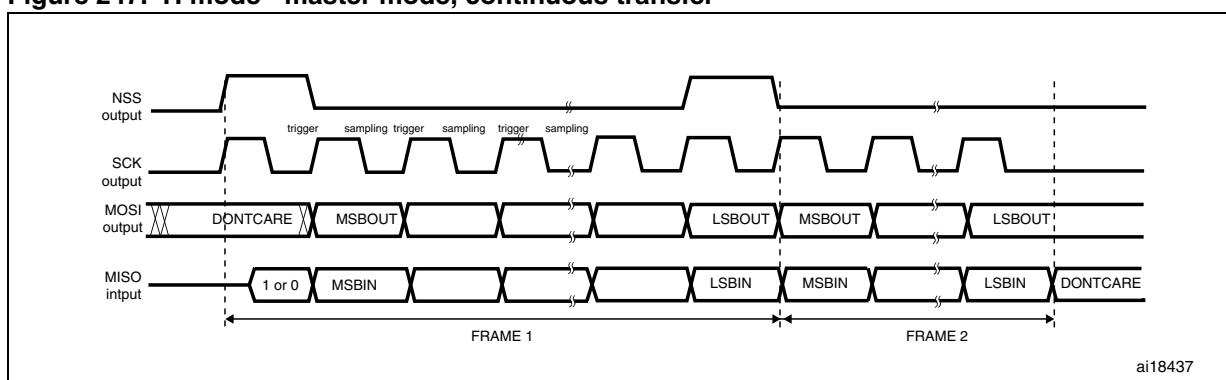


Figure 247. TI mode - master mode, continuous transfer



25.3.4 Configuring the SPI for simplex communication

The SPI is capable of operating in simplex mode in 2 configurations.

- 1 clock and 1 bidirectional data wire
- 1 clock and 1 data wire (receive-only or transmit-only)

1 clock and 1 bidirectional data wire (BIDIMODE=1)

This mode is enabled by setting the BIDIMODE bit in the SPI_CR1 register. In this mode SCK is used for the clock and MOSI in master or MISO in slave mode is used for data communication. The transfer direction (Input/Output) is selected by the BIDIOE bit in the SPI_CR1 register. When this bit is 1, the data line is output otherwise it is input.

1 clock and 1 unidirectional data wire (BIDIMODE=0)

In this mode, the application can use the SPI either in transmit-only mode or in receive-only mode.

- Transmit-only mode is similar to full-duplex mode (BIDIMODE=0, RXONLY=0): the data are transmitted on the transmit pin (MOSI in master mode or MISO in slave mode) and the receive pin (MISO in master mode or MOSI in slave mode) can be used as a general-purpose IO. In this case, the application just needs to ignore the Rx buffer (if the data register is read, it does not contain the received value).
- In receive-only mode, the application can disable the SPI output function by setting the RXONLY bit in the SPI_CR2 register. In this case, it frees the transmit IO pin (MOSI in master mode or MISO in slave mode), so it can be used for other purposes.

To start the communication in receive-only mode, configure and enable the SPI:

- In master mode, the communication starts immediately and stops when the SPE bit is cleared and the current reception stops. There is no need to read the BSY flag in this mode. It is always set when an SPI communication is ongoing.
- In slave mode, the SPI continues to receive as long as the NSS is pulled down (or the SSI bit is cleared in NSS software mode) and the SCK is running.

25.3.5 Data transmission and reception procedures

Rx and Tx buffers

In reception, data are received and then stored into an internal Rx buffer while in transmission, data are first stored into an internal Tx buffer before being transmitted.

A read access of the SPI_DR register returns the Rx buffered value whereas a write access to the SPI_DR stores the written data into the Tx buffer.

Start sequence in master mode

- In full-duplex (BIDIMODE=0 and RXONLY=0)
 - The sequence begins when data are written into the SPI_DR register (Tx buffer).
 - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
 - At the same time, the received data on the MISO pin is shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
- In unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
 - The sequence begins as soon as SPE=1
 - Only the receiver is activated and the received data on the MISO pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
- In bidirectional mode, when transmitting (BIDIMODE=1 and BIDIOE=1)
 - The sequence begins when data are written into the SPI_DR register (Tx buffer).
 - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
 - No data are received.
- In bidirectional mode, when receiving (BIDIMODE=1 and BIDIOE=0)
 - The sequence begins as soon as SPE=1 and BIDIOE=0.
 - The received data on the MOSI pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
 - The transmitter is not activated and no data are shifted out serially to the MOSI pin.

Start sequence in slave mode

- In full-duplex mode (BIDIMODE=0 and RXONLY=0)
 - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The 7 remaining bits are loaded into the shift register.
 - At the same time, the data are parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission, and then shifted out serially to the MISO pin. The software must have written the data to be sent before the SPI master device initiates the transfer.
- In unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
 - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The 7 remaining bits are loaded into the shift register.
 - The transmitter is not activated and no data are shifted out serially to the MISO pin.
- In bidirectional mode, when transmitting (BIDIMODE=1 and BIDIOE=1)
 - The sequence begins when the slave device receives the clock signal and the first bit in the Tx buffer is transmitted on the MISO pin.
 - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MISO pin. The software must have written the data to be sent before the SPI master device initiates the transfer.
 - No data are received.

- In bidirectional mode, when receiving (BIDIMODE=1 and BIDIOE=0)
 - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MISO pin.
 - The received data on the MISO pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
 - The transmitter is not activated and no data are shifted out serially to the MISO pin.

Handling data transmission and reception

The TXE flag (Tx buffer empty) is set when the data are transferred from the Tx buffer to the shift register. It indicates that the internal Tx buffer is ready to be loaded with the next data. An interrupt can be generated if the TXIE bit in the SPI_CR2 register is set. Clearing the TXE bit is performed by writing to the SPI_DR register.

Note: *The software must ensure that the TXE flag is set to 1 before attempting to write to the Tx buffer. Otherwise, it overwrites the data previously written to the Tx buffer.*

The RXNE flag (Rx buffer not empty) is set on the last sampling clock edge, when the data are transferred from the shift register to the Rx buffer. It indicates that data are ready to be read from the SPI_DR register. An interrupt can be generated if the RXNEIE bit in the SPI_CR2 register is set. Clearing the RXNE bit is performed by reading the SPI_DR register.

For some configurations, the BSY flag can be used during the last data transfer to wait until the completion of the transfer.

Full-duplex transmit and receive procedure in master or slave mode (BIDIMODE=0 and RXONLY=0)

The software has to follow this procedure to transmit and receive data (see [Figure 248](#) and [Figure 249](#)):

1. Enable the SPI by setting the SPE bit to 1.
2. Write the first data item to be transmitted into the SPI_DR register (this clears the TXE flag).
3. Wait until TXE=1 and write the second data item to be transmitted. Then wait until RXNE=1 and read the SPI_DR to get the first received data item (this clears the RXNE bit). Repeat this operation for each data item to be transmitted/received until the n-1 received data.
4. Wait until RXNE=1 and read the last received data.
5. Wait until TXE=1 and then wait until BSY=0 before disabling the SPI.

This procedure can also be implemented using dedicated interrupt subroutines launched at each rising edges of the RXNE or TXE flag.

Figure 248. TXE/RXNE/BSY behavior in Master / full-duplex mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers

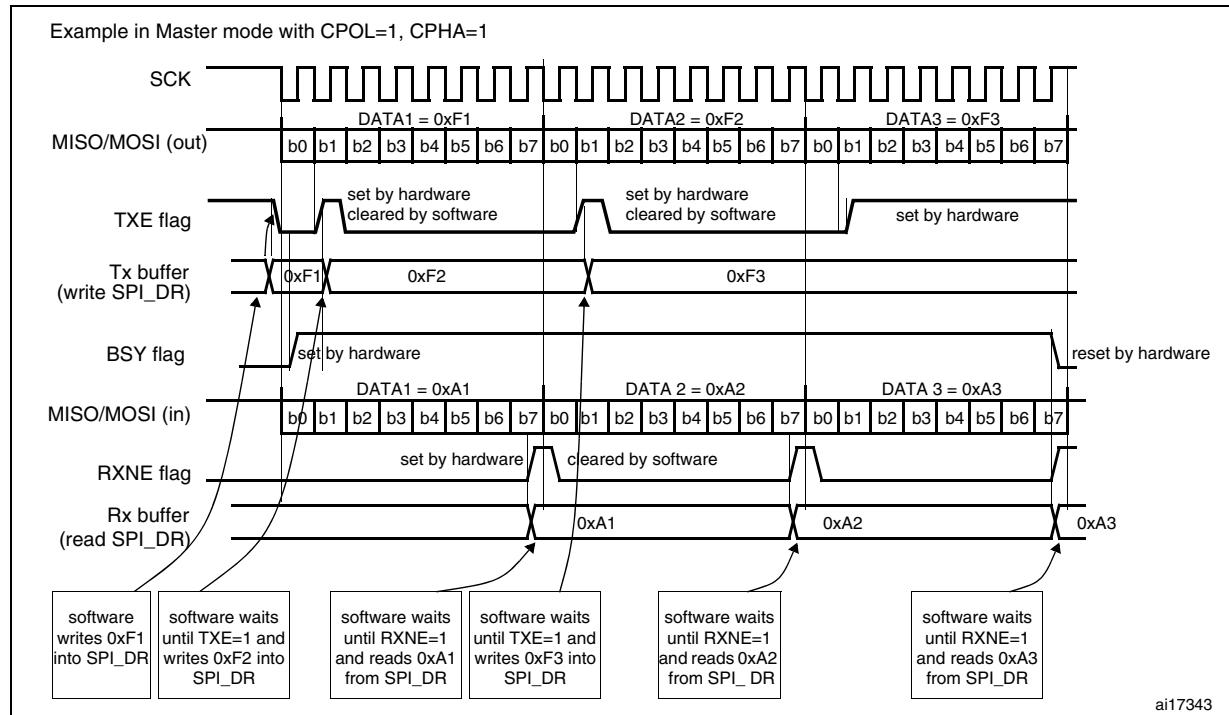
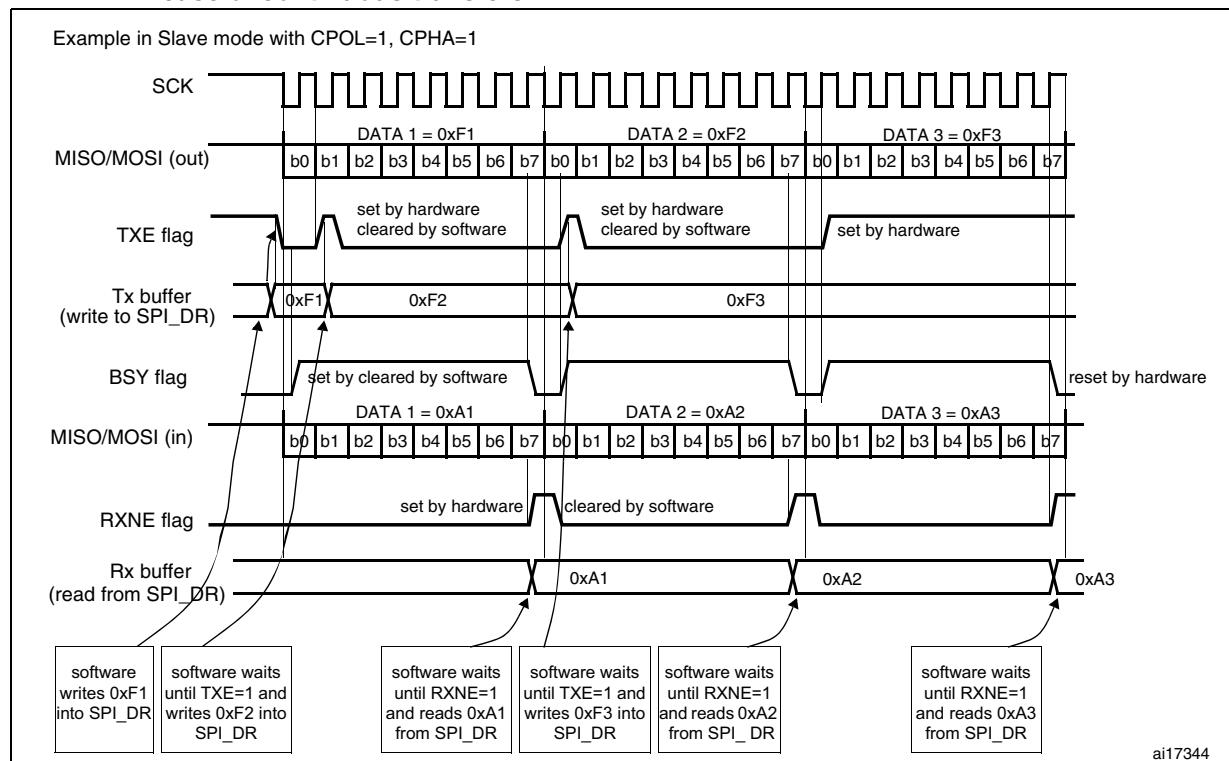


Figure 249. TXE/RXNE/BSY behavior in Slave / full-duplex mode (BIDIMODE=0, RXONLY=0) in the case of continuous transfers



Transmit-only procedure (BIDIMODE=0 RXONLY=0)

In this mode, the procedure can be reduced as described below and the BSY bit can be used to wait until the completion of the transmission (see [Figure 250](#) and [Figure 251](#)).

1. Enable the SPI by setting the SPE bit to 1.
2. Write the first data item to send into the SPI_DR register (this clears the TXE bit).
3. Wait until TXE=1 and write the next data item to be transmitted. Repeat this step for each data item to be transmitted.
4. After writing the last data item into the SPI_DR register, wait until TXE=1, then wait until BSY=0, this indicates that the transmission of the last data is complete.

This procedure can be also implemented using dedicated interrupt subroutines launched at each rising edge of the TXE flag.

- Note:**
- 1 *During discontinuous communications, there is a 2 APB clock period delay between the write operation to SPI_DR and the BSY bit setting. As a consequence, in transmit-only mode, it is mandatory to wait first until TXE is set and then until BSY is cleared after writing the last data.*
 - 2 *After transmitting two data items in transmit-only mode, the OVR flag is set in the SPI_SR register since the received data are never read.*

Figure 250. TXE/BSY behavior in Master transmit-only mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers

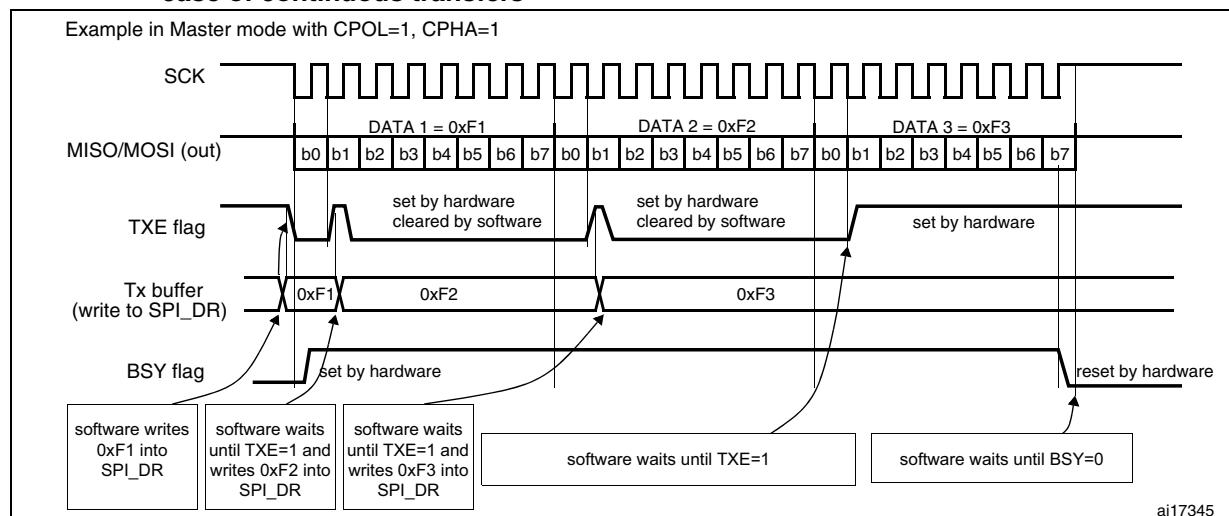
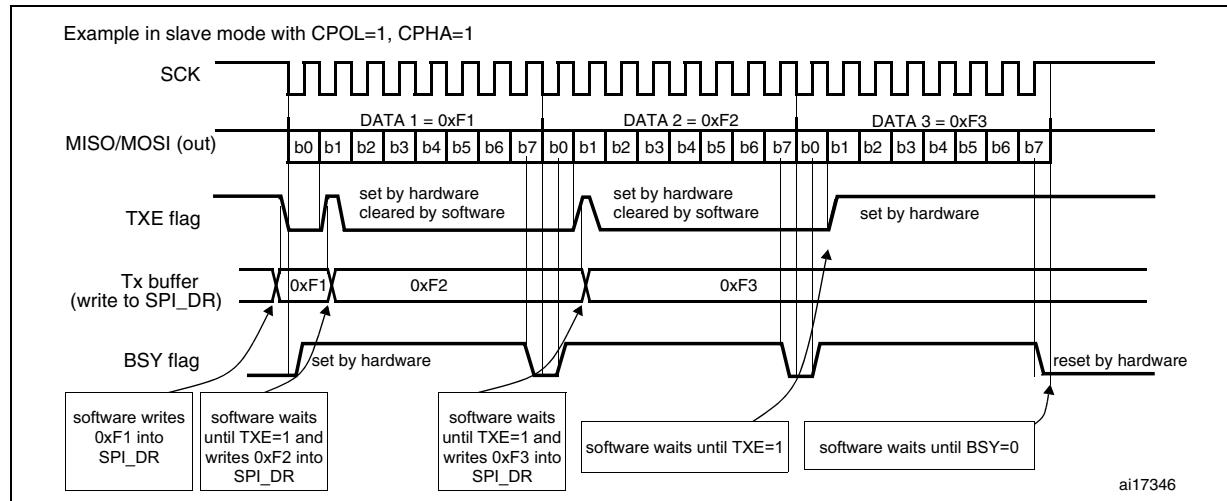


Figure 251. TXE/BSY in Slave transmit-only mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers



Bidirectional transmit procedure (BIDIMODE=1 and BIDIOE=1)

In this mode, the procedure is similar to the procedure in Transmit-only mode except that the BIDIMODE and BIDIOE bits both have to be set in the SPI_CR2 register before enabling the SPI.

Unidirectional receive-only procedure (BIDIMODE=0 and RXONLY=1)

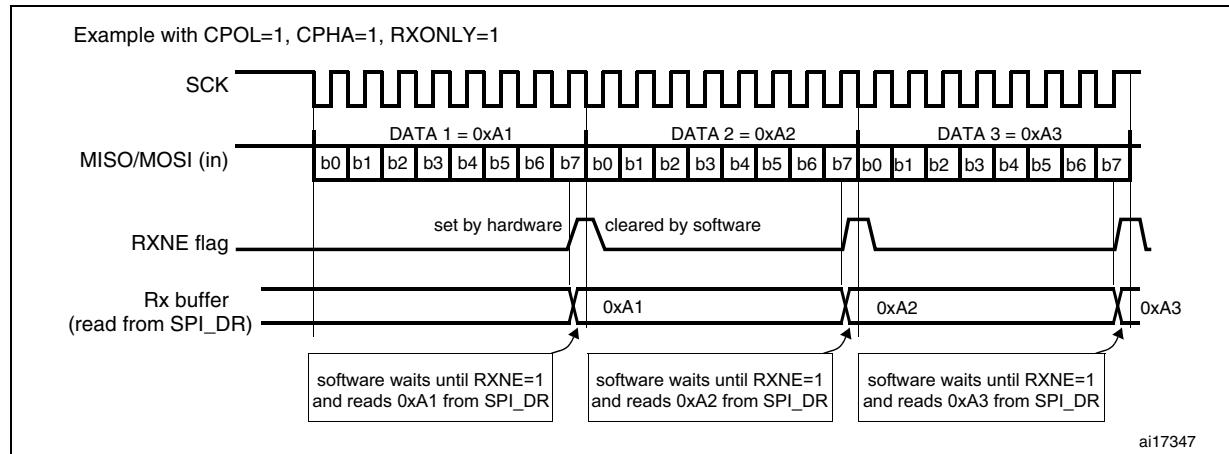
In this mode, the procedure can be reduced as described below (see [Figure 252](#)):

1. Set the RXONLY bit in the SPI_CR2 register.
2. Enable the SPI by setting the SPE bit to 1:
 - a) In master mode, this immediately activates the generation of the SCK clock, and data are serially received until the SPI is disabled (SPE=0).
 - b) In slave mode, data are received when the SPI master device drives NSS low and generates the SCK clock.
3. Wait until RXNE=1 and read the SPI_DR register to get the received data (this clears the RXNE bit). Repeat this operation for each data item to be received.

This procedure can also be implemented using dedicated interrupt subroutines launched at each rising edge of the RXNE flag.

Note: If it is required to disable the SPI after the last transfer, follow the recommendation described in [Section 25.3.8: Disabling the SPI on page 679](#).

Figure 252. RXNE behavior in receive-only mode (BIDIMODE=0 and RXONLY=1) in the case of continuous transfers



Bidirectional receive procedure (BIDIMODE=1 and BIDIOE=0)

In this mode, the procedure is similar to the Receive-only mode procedure except that the BIDIMODE bit has to be set and the BIDIOE bit cleared in the SPI_CR2 register before enabling the SPI.

Continuous and discontinuous transfers

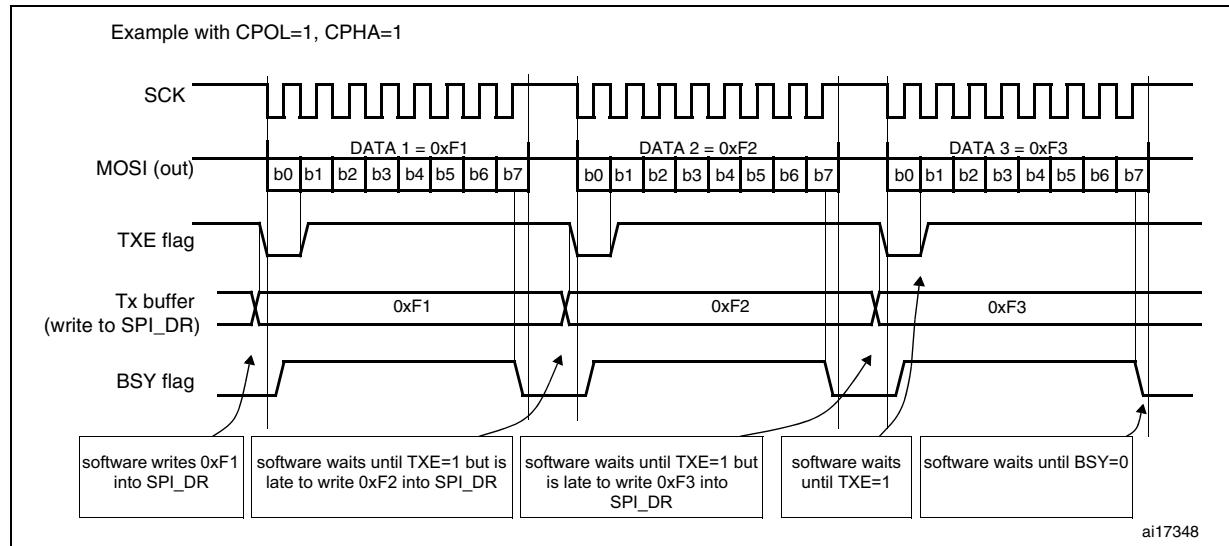
When transmitting data in master mode, if the software is fast enough to detect each rising edge of TXE (or TXE interrupt) and to immediately write to the SPI_DR register before the ongoing data transfer is complete, the communication is said to be continuous. In this case, there is no discontinuity in the generation of the SPI clock between each data item and the BSY bit is never cleared between each data transfer.

On the contrary, if the software is not fast enough, this can lead to some discontinuities in the communication. In this case, the BSY bit is cleared between each data transmission (see [Figure 253](#)).

In Master receive-only mode (RXONLY=1), the communication is always continuous and the BSY flag is always read at 1.

In slave mode, the continuity of the communication is decided by the SPI master device. In any case, even if the communication is continuous, the BSY flag goes low between each transfer for a minimum duration of one SPI clock cycle (see [Figure 251](#)).

Figure 253. TXE/BSY behavior when transmitting (BIDIRMODE=0 and RXONLY=0) in the case of discontinuous transfers



25.3.6 CRC calculation

A CRC calculator has been implemented for communication reliability. Separate CRC calculators are implemented for transmitted data and received data. The CRC is calculated using a programmable polynomial serially on each bit. It is calculated on the sampling clock edge defined by the CPHA and CPOL bits in the SPI_CR1 register.

Note: This SPI offers two kinds of CRC calculation standard which depend directly on the data frame format selected for the transmission and/or reception: 8-bit data (CR8) and 16-bit data (CRC16).

CRC calculation is enabled by setting the CRCEN bit in the SPI_CR1 register. This action resets the CRC registers (SPI_RXCRCR and SPI_TXCRCR). In full duplex or transmitter only mode, when the transfers are managed by the software (CPU mode), it is necessary to write the bit CRCNEXT immediately after the last data to be transferred is written to the SPI_DR. At the end of this last data transfer, the SPI_TXCRCR value is transmitted.

In receive only mode and when the transfers are managed by software (CPU mode), it is necessary to write the CRCNEXT bit after the second last data has been received. The CRC is received just after the last data reception and the CRC check is then performed.

At the end of data and CRC transfers, the CRCERR flag in the SPI_SR register is set if corruption occurs during the transfer.

If data are present in the TX buffer, the CRC value is transmitted only after the transmission of the data byte. During CRC transmission, the CRC calculator is switched off and the register value remains unchanged.

Note: Please refer to the product datasheet for availability of this feature.

SPI communication using the CRC is possible through the following procedure:

1. Program the CPOL, CPHA, LSBFirst, BR, SSM, SSI and MSTR values.
2. Program the polynomial in the SPI_CRCPR register.
3. Enable the CRC calculation by setting the CRCEN bit in the SPI_CR1 register. This also clears the SPI_RXCRCR and SPI_TXCRCR registers.
4. Enable the SPI by setting the SPE bit in the SPI_CR1 register.
5. Start the communication and sustain the communication until all but one byte or half-word have been transmitted or received.
 - In full duplex or transmitter-only mode, when the transfers are managed by software, when writing the last byte or half word to the Tx buffer, set the CRCNEXT bit in the SPI_CR1 register to indicate that the CRC will be transmitted after the transmission of the last byte.
 - In receiver only mode, set the bit CRCNEXT just after the reception of the second to last data to prepare the SPI to enter in CRC Phase at the end of the reception of the last data. CRC calculation is frozen during the CRC transfer.
6. After the transfer of the last byte or half word, the SPI enters the CRC transfer and check phase. In full duplex mode or receiver-only mode, the received CRC is compared to the SPI_RXCRCR value. If the value does not match, the CRCERR flag in SPI_SR is set and an interrupt can be generated when the ERRIE bit in the SPI_CR2 register is set.

Note:

When the SPI is in slave mode, be careful to enable CRC calculation only when the clock is stable, that is, when the clock is in the steady state. If not, a wrong CRC calculation may be done. In fact, the CRC is sensitive to the SCK slave input clock as soon as CRCEN is set, and this, whatever the value of the SPE bit.

With high bitrate frequencies, be careful when transmitting the CRC. As the number of used CPU cycles has to be as low as possible in the CRC transfer phase, it is forbidden to call software functions in the CRC transmission sequence to avoid errors in the last data and CRC reception. In fact, CRCNEXT bit has to be written before the end of the transmission/reception of the last data.

For high bit rate frequencies, it is advised to use the DMA mode to avoid the degradation of the SPI speed performance due to CPU accesses impacting the SPI bandwidth.

When the STM32F40x and STM32F41x are configured as slaves and the NSS hardware mode is used, the NSS pin needs to be kept low between the data phase and the CRC phase.

When the SPI is configured in slave mode with the CRC feature enabled, CRC calculation takes place even if a high level is applied on the NSS pin. This may happen for example in case of a multislave environment where the communication master addresses slaves alternately.

Between a slave deselection (high level on NSS) and a new slave selection (low level on NSS), the CRC value should be cleared on both master and slave sides in order to resynchronize the master and slave for their respective CRC calculation.

To clear the CRC, follow the procedure below:

1. Disable SPI (SPE = 0)
2. Clear the CRCEN bit
3. Set the CRCEN bit
4. Enable the SPI (SPE = 1)

25.3.7 Status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

When it is set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can be loaded into the buffer. The TXE flag is cleared when writing to the SPI_DR register.

Rx buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the Rx buffer. It is cleared when SPI_DR is read.

BSY flag

This BSY flag is set and cleared by hardware (writing to this flag has no effect). The BSY flag indicates the state of the communication layer of the SPI.

When BSY is set, it indicates that the SPI is busy communicating. There is one exception in master mode / bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software wants to disable the SPI and enter Halt mode (or disable the peripheral clock). This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is also useful to avoid write collisions in a multimaster system.

The BSY flag is set when a transfer starts, with the exception of master mode / bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0).

It is cleared:

- when a transfer is finished (except in master mode if the communication is continuous)
- when the SPI is disabled
- when a master mode fault occurs (MODF=1)

When communication is not continuous, the BSY flag is low between each communication.

When communication is continuous:

- in master mode, the BSY flag is kept high during all the transfers
- in slave mode, the BSY flag goes low for one SPI clock cycle between each transfer

Note: Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.

25.3.8 Disabling the SPI

When a transfer is terminated, the application can stop the communication by disabling the SPI peripheral. This is done by clearing the SPE bit.

For some configurations, disabling the SPI and entering the Halt mode while a transfer is ongoing can cause the current transfer to be corrupted and/or the BSY flag might become unreliable.

To avoid any of those effects, it is recommended to respect the following procedure when disabling the SPI:

In master or slave full-duplex mode (BIDIMODE=0, RXONLY=0)

1. Wait until RXNE=1 to receive the last data
2. Wait until TXE=1
3. Then wait until BSY=0
4. Disable the SPI (SPE=0) and, eventually, enter the Halt mode (or disable the peripheral clock)

In master or slave unidirectional transmit-only mode (BIDIMODE=0, RXONLY=0) or bidirectional transmit mode (BIDIMODE=1, BIDIOE=1)

After the last data is written into the SPI_DR register:

1. Wait until TXE=1
2. Then wait until BSY=0
3. Disable the SPI (SPE=0) and, eventually, enter the Halt mode (or disable the peripheral clock)

In master unidirectional receive-only mode (MSTR=1, BIDIMODE=0, RXONLY=1) or bidirectional receive mode (MSTR=1, BIDIMODE=1, BIDIOE=0)

This case must be managed in a particular way to ensure that the SPI does not initiate a new transfer. The sequence below is valid only for SPI Motorola configuration (FRF bit set to 0):

1. Wait for the second to last occurrence of RXNE=1 (n-1)
2. Then wait for one SPI clock cycle (using a software loop) before disabling the SPI (SPE=0)
3. Then wait for the last RXNE=1 before entering the Halt mode (or disabling the peripheral clock)

When the SPI is configured in TI mode (Bit FRF set to 1), the following procedure has to be respected to avoid generating an undesired pulse on NSS when the SPI is disabled:

1. Wait for the second to last occurrence of RXNE = 1 (n-1).
2. Disable the SPI (SPE = 0) in the following window frame using a software loop:
 - After at least one SPI clock cycle,
 - Before the beginning of the LSB data transfer.

Note: In master bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0), the BSY flag is kept low during transfers.

In slave receive-only mode (MSTR=0, BIDIMODE=0, RXONLY=1) or bidirectional receive mode (MSTR=0, BIDIMODE=1, BDOE=0)

1. You can disable the SPI (write SPE=1) at any time: the current transfer will complete before the SPI is effectively disabled
2. Then, if you want to enter the Halt mode, you must first wait until BSY = 0 before entering the Halt mode (or disabling the peripheral clock).

25.3.9 SPI communication using DMA (direct memory addressing)

To operate at its maximum speed, the SPI needs to be fed with the data for transmission and the data received on the Rx buffer should be read to avoid overrun. To facilitate the transfers, the SPI features a DMA capability implementing a simple request/acknowledge protocol.

A DMA access is requested when the enable bit in the SPI_CR2 register is enabled.

Separate requests must be issued to the Tx and Rx buffers (see [Figure 254](#) and [Figure 255](#)):

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPI_DR register (this clears the TXE flag).
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPI_DR register (this clears the RXNE flag).

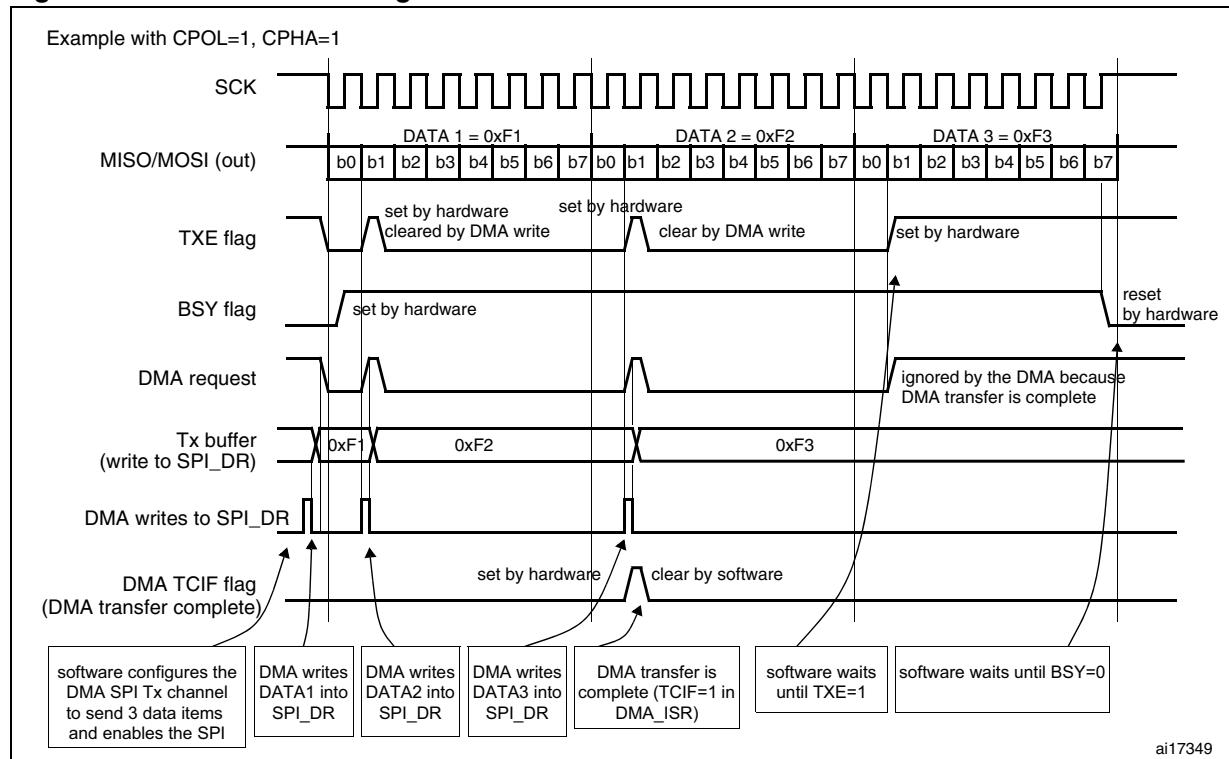
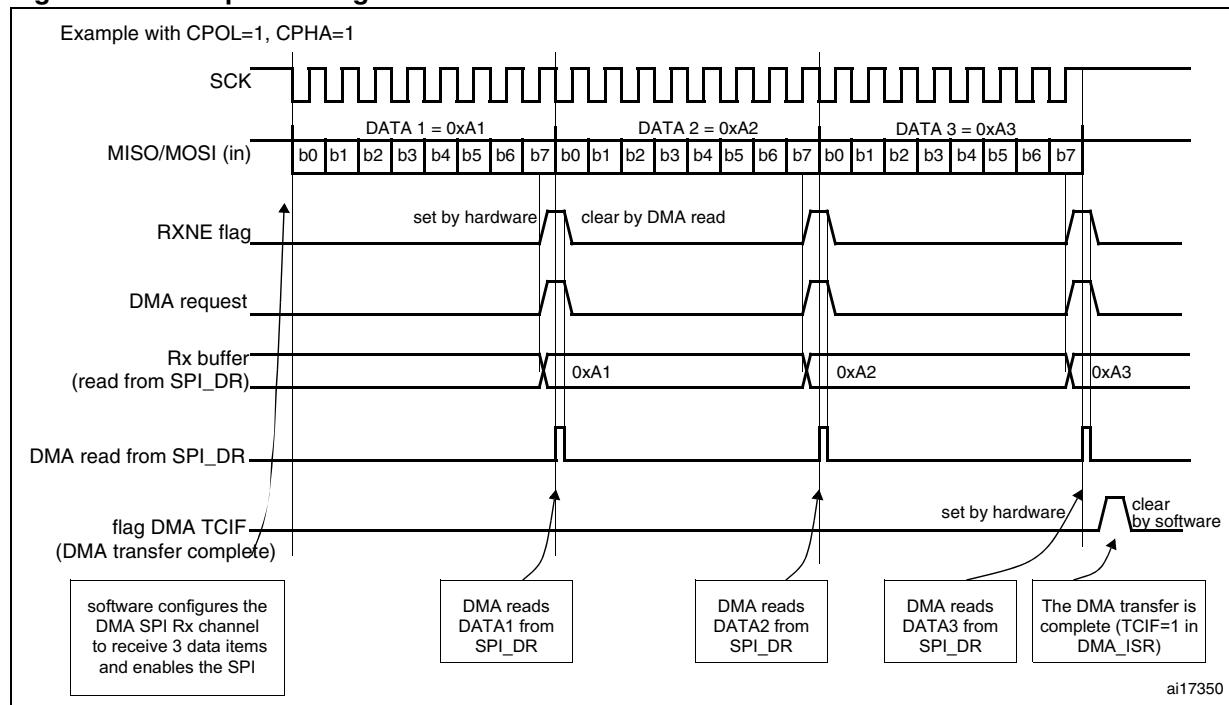
When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received are not read.

When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (flag TCIF is set in the DMA_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. The software must first wait until TXE=1 and then until BSY=0.

Note:

During discontinuous communications, there is a 2 APB clock period delay between the write operation to SPI_DR and the BSY bit setting. As a consequence, it is mandatory to wait first until TXE=1 and then until BSY=0 after writing the last data.

Figure 254. Transmission using DMA**Figure 255. Reception using DMA**

DMA capability with CRC

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication are automatic that is without using the bit CRCNEXT. After the CRC reception, the CRC must be read in the SPI_DR register to clear the RXNE flag.

At the end of data and CRC transfers, the CRCERR flag in SPI_SR is set if corruption occurs during the transfer.

25.3.10 Error flags

Master mode fault (MODF)

Master mode fault occurs when the master device has its NSS pin pulled low (in NSS hardware mode) or SSI bit low (in NSS software mode), this automatically sets the MODF bit. Master mode fault affects the SPI peripheral in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPI_SR register while the MODF bit is set.
2. Then write to the SPI_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence.

As a security, hardware does not allow the setting of the SPE and MSTR bits while the MODF bit is set.

In a slave device the MODF bit cannot be set. However, in a multimaster configuration, the device can be in slave mode with this MODF bit set. In this case, the MODF bit indicates that there might have been a multimaster conflict for system control. An interrupt routine can be used to recover cleanly from this state by performing a reset or returning to a default state.

Overrun condition

An overrun condition occurs when the master device has sent data bytes and the slave device has not cleared the RXNE bit resulting from the previous data byte transmitted.

When an overrun condition occurs:

- the OVR bit is set and an interrupt is generated if the ERRIE bit is set.

In this case, the receiver buffer contents will not be updated with the newly received data from the master device. A read from the SPI_DR register returns this byte. All other subsequently transmitted bytes are lost.

Clearing the OVR bit is done by a read from the SPI_DR register followed by a read access to the SPI_SR register.

CRC error

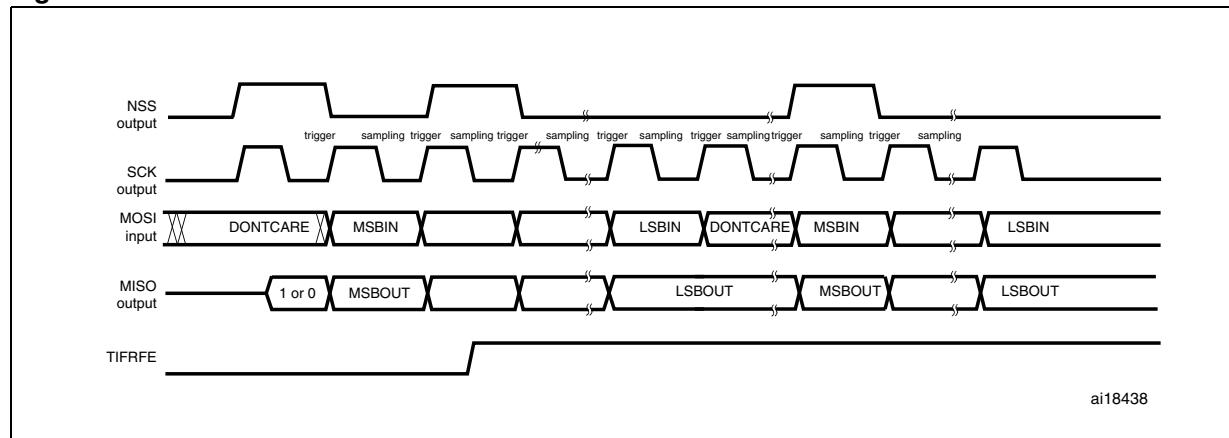
This flag is used to verify the validity of the value received when the CRCEN bit in the SPI_CR1 register is set. The CRCERR flag in the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRCR value.

TI mode frame format error

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is acting in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRFE flag is set in the SPI_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 lost of two data bytes.

The TIFRFE flag is cleared when SPI_SR register is read. If the bit ERRIE is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no more guaranteed and communications should be reinitiated by the master when the slave SPI is re-enabled.

Figure 256. TI mode frame format error detection



25.3.11 SPI interrupts

Table 99. SPI interrupt requests

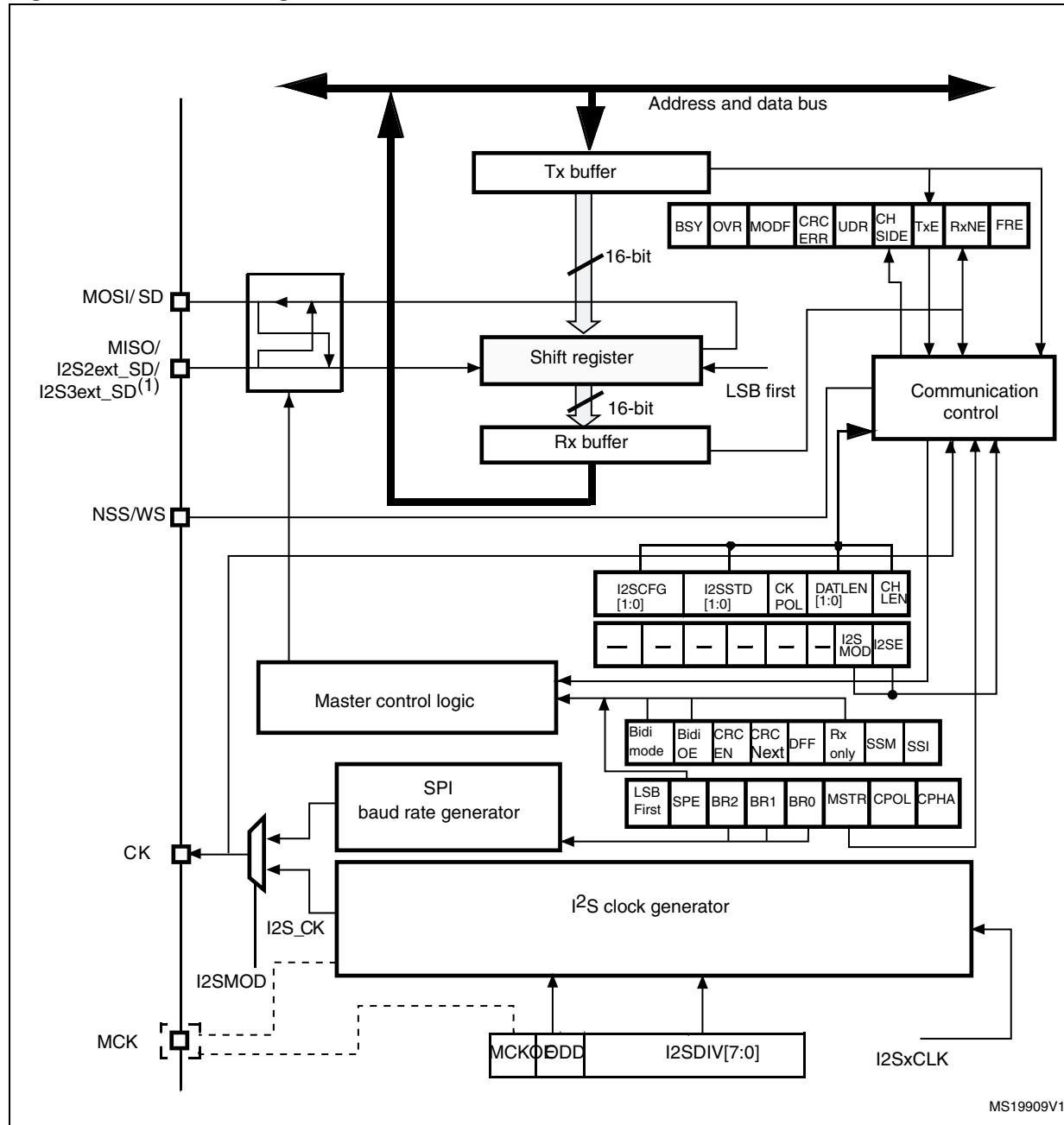
Interrupt event	Event flag	Enable Control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
CRC error flag	CRCERR	
TI frame format error	TIFRFE	ERRIE

25.4 I²S functional description

25.4.1 I²S general description

The block diagram of the I²S is shown in [Figure 257](#).

Figure 257. I²S block diagram



1. I²S2ext_SD and I²S3ext_SD are the extended SD pins that control the I²S full duplex mode.

The SPI could function as an audio I²S interface when the I²S capability is enabled (by setting the I2SMOD bit in the SPI_I2SCFGR register). This interface uses almost 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 simplex 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 could 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 SPI_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 SPI_I2SPR and the other one is a generic I²S configuration register SPI_I2SCFGR (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPI_CR1 register and all CRC registers are not used in the I²S mode. Likewise, the SSOE bit in the SPI_CR2 register and the MODF and CRCERR bits in the SPI_SR are not used.

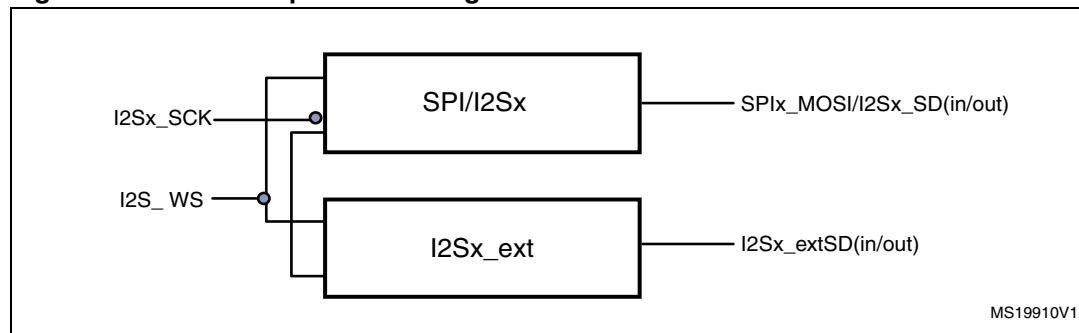
The I²S uses the same SPI register for data transfer (SPI_DR) in 16-bit wide mode.

25.4.2 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 258](#)). 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 258. 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.

25.4.3 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 the transmission and the reception. So, it is up to the software to write into the data register the adequate value corresponding to the considered channel side, or to read the data from the data register and to identify the corresponding channel by checking the CHSIDE bit in the SPI_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 16-bit frame
- 16-bit data packed in 32-bit frame
- 24-bit data packed in 32-bit frame
- 32-bit data packed in 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).

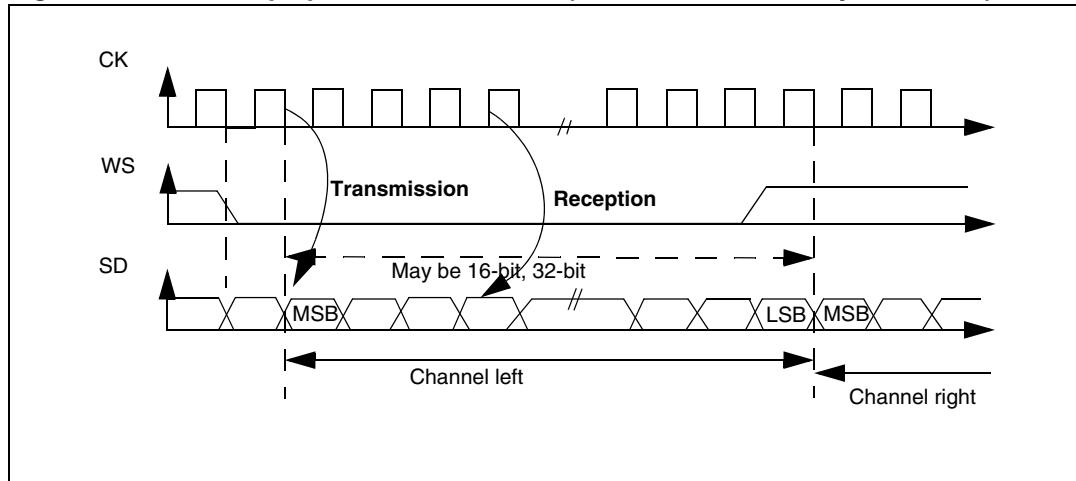
The 24-bit and 32-bit data frames need two CPU read or write operations to/from the SPI_DR or two DMA operations if the DMA is preferred for the application. For 24-bit data frame specifically, the 8 nonsignificant bits are extended to 32 bits with 0-bits (by hardware).

For all data formats and communication standards, the most significant bit is always sent first (MSB first).

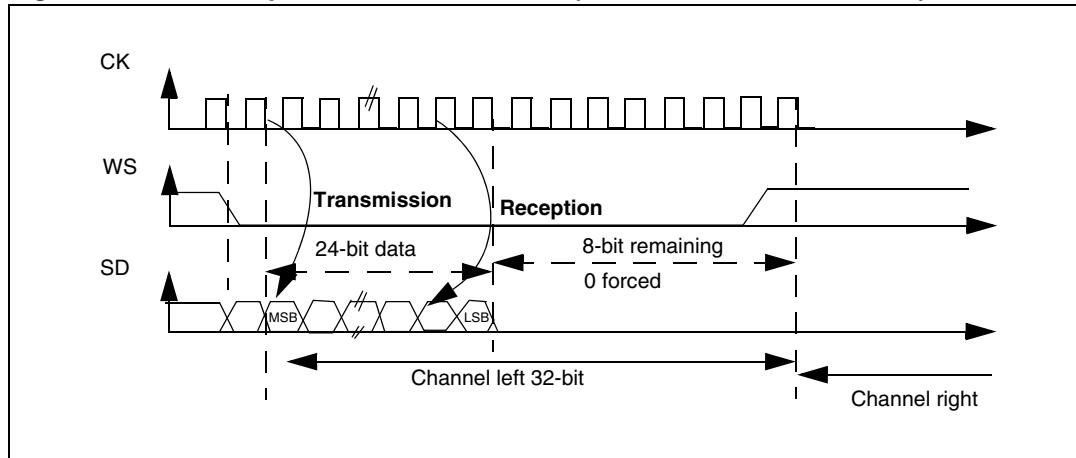
The I²S interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPI_I2SCFGR register.

I²S Phillips 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.

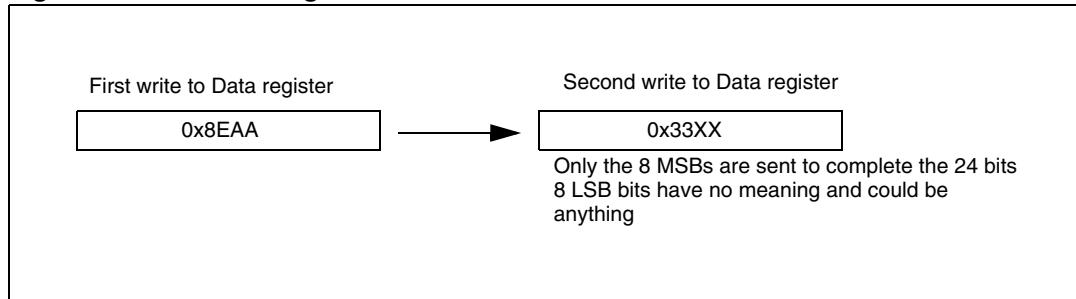
Figure 259. I²S Phillips protocol waveforms (16/32-bit full accuracy, CPOL = 0)

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.

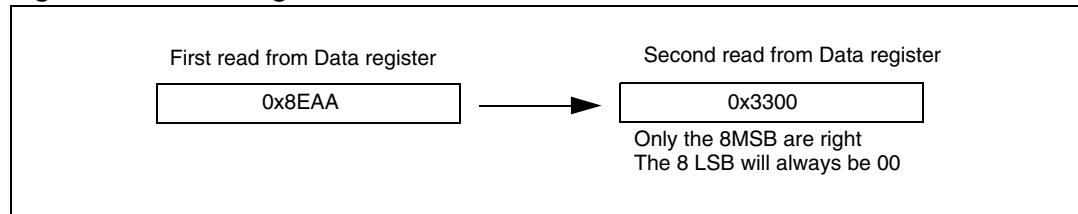
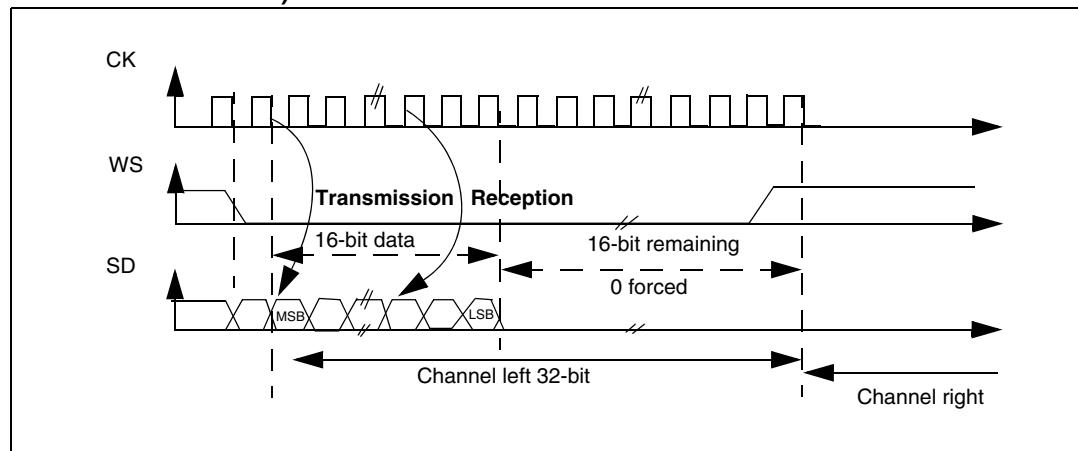
Figure 260. I²S Phillips standard waveforms (24-bit frame with CPOL = 0)

This mode needs two write or read operations to/from the SPI_DR.

- In transmission mode:
if 0x8EAA33 has to be sent (24-bit):

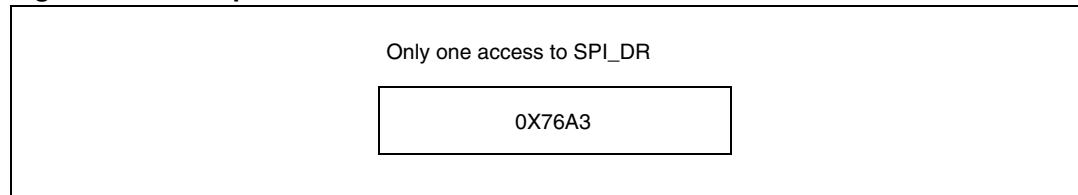
Figure 261. Transmitting 0x8EAA33

- In reception mode:
if data 0x8EAA33 is received:

Figure 262. Receiving 0x8EAA33**Figure 263. I²S Phillips 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 SPI_DR 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 264](#) is required.

Figure 264. Example

For transmission, each time an MSB is written to SPI_DR, the TXE flag is set and its interrupt, if allowed, is generated to load SPI_DR with the new value to send. This takes place even if 0x0000 have not yet been sent because it is done by hardware.

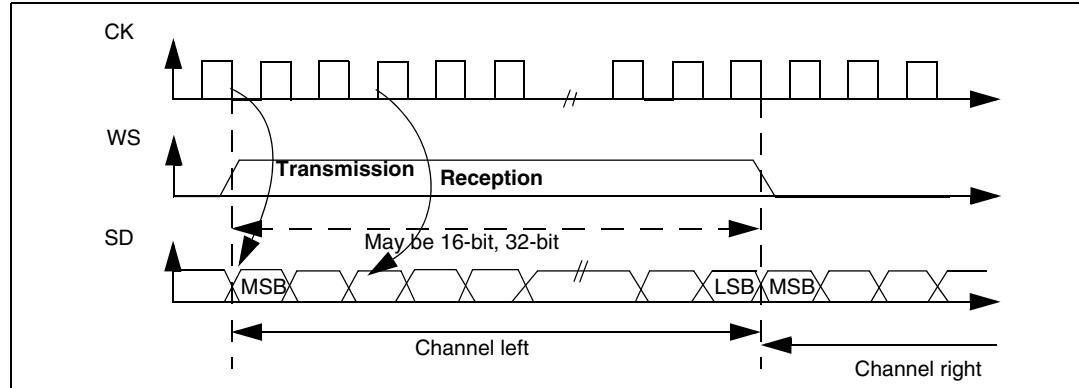
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

In this way, more time is provided between two write or read operations, which prevents underrun or overrun conditions (depending on the direction of the data transfer).

MSB justified standard

For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

Figure 265. 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 266. MSB Justified 24-bit frame length with CPOL = 0

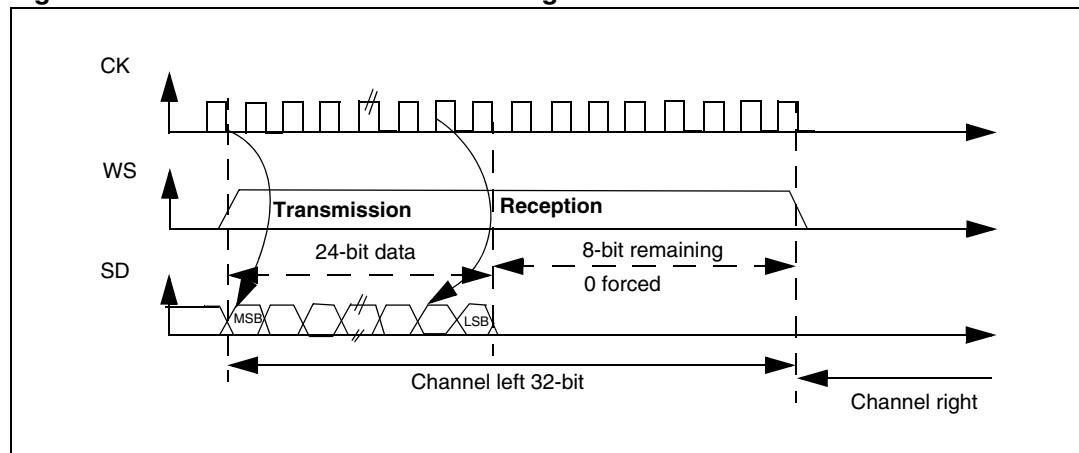
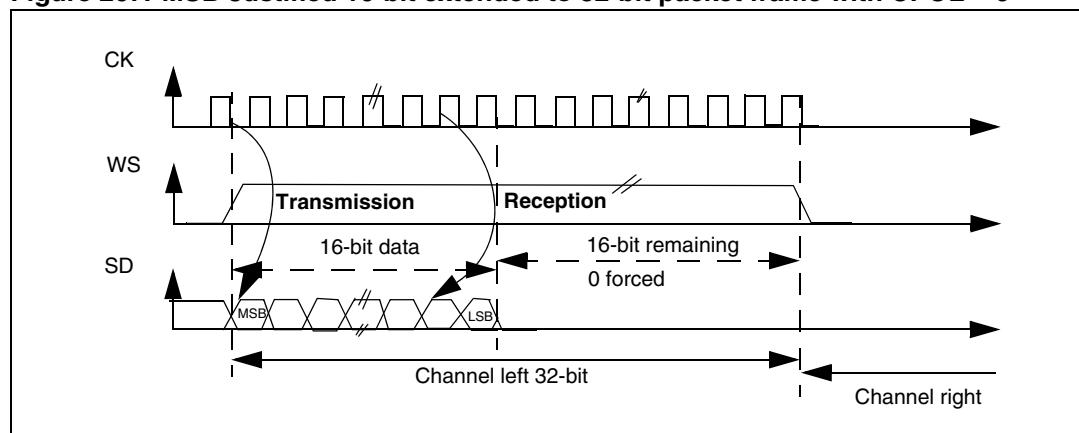


Figure 267. MSB Justified 16-bit extended to 32-bit packet frame with CPOL = 0



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 268. LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0

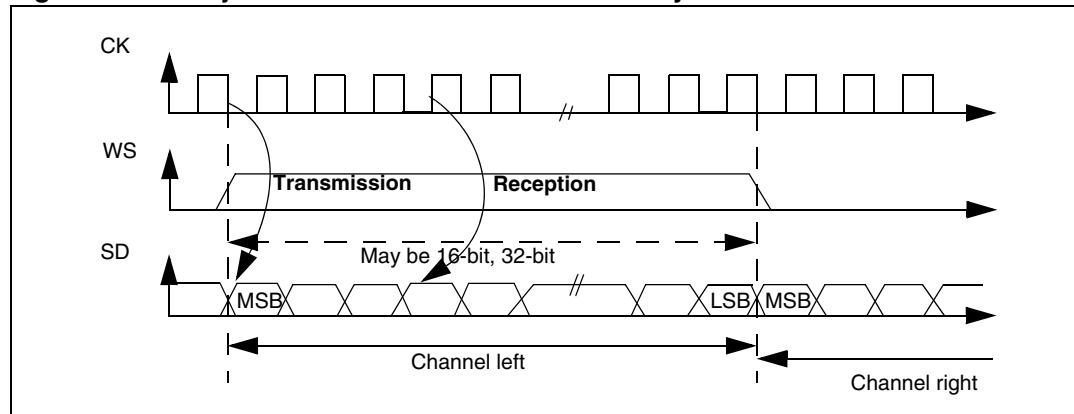
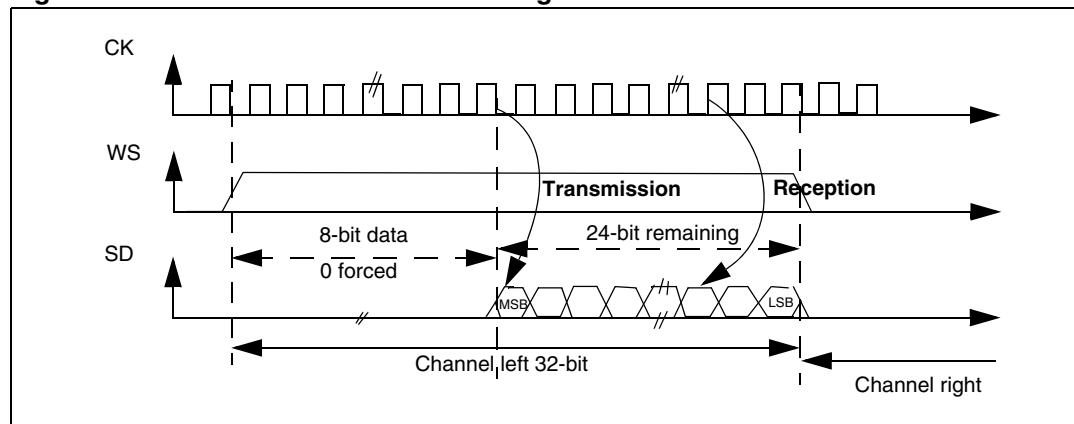


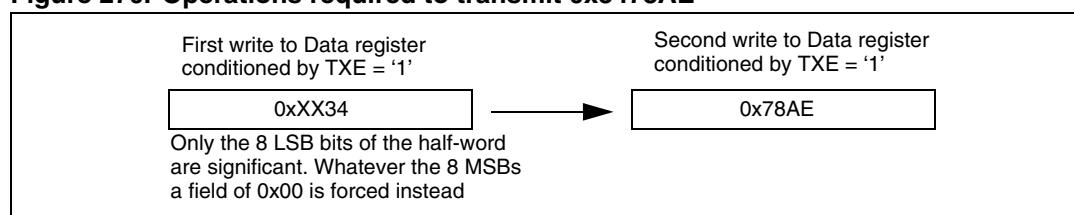
Figure 269. LSB Justified 24-bit frame length with CPOL = 0



- In transmission mode:

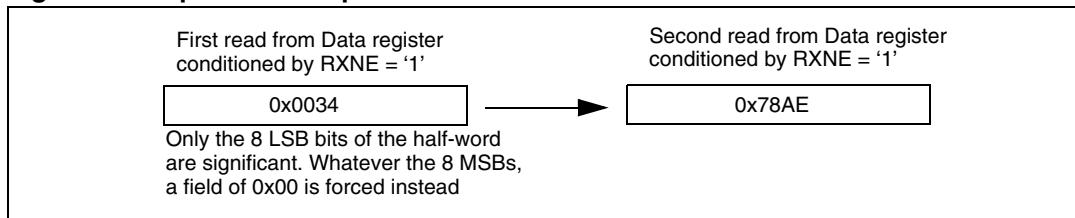
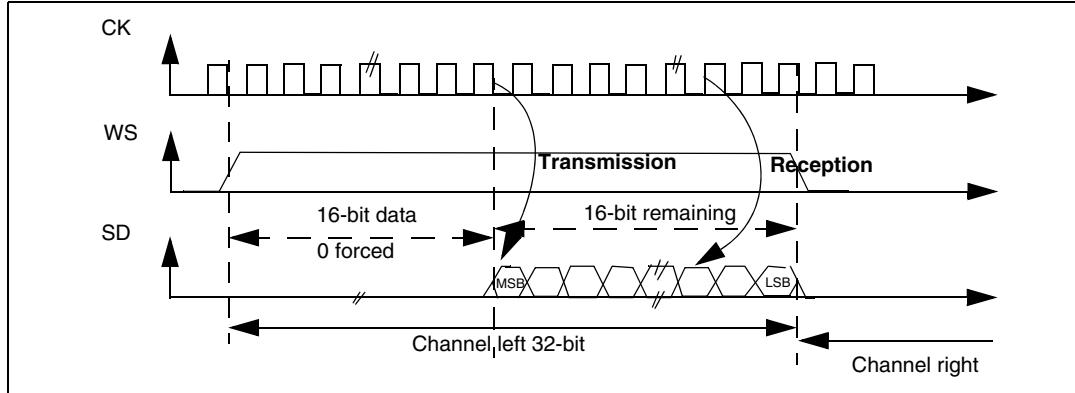
If data 0x3478AE have to be transmitted, two write operations to the SPI_DR register are required from software or by DMA. The operations are shown below.

Figure 270. Operations required to transmit 0x3478AE



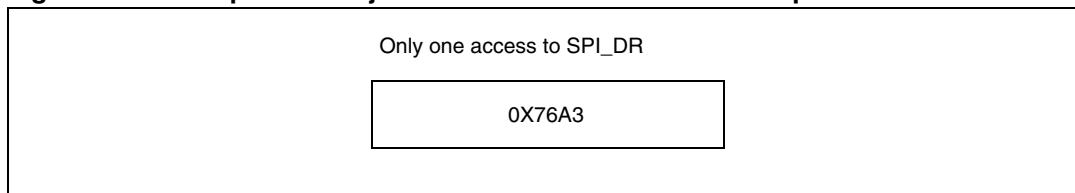
- In reception mode:

If data 0x3478AE are received, two successive read operations from SPI_DR are required on each RXNE event.

Figure 271. Operations required to receive 0x3478AE**Figure 272. 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 SPI_DR 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 273](#) is required.

Figure 273. Example of LSB justified 16-bit extended to 32-bit packet frame

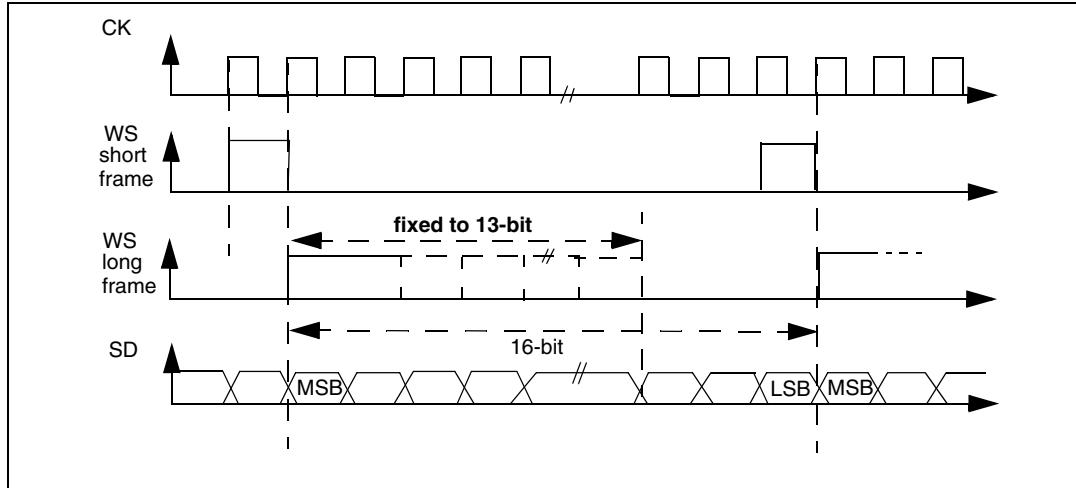
In transmission mode, when TXE is asserted, the application has to write the data to be transmitted (in this case 0x76A3). The 0x000 field is transmitted first (extension on 32-bit). TXE is asserted again as soon as the effective data (0x76A3) is sent on SD.

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.

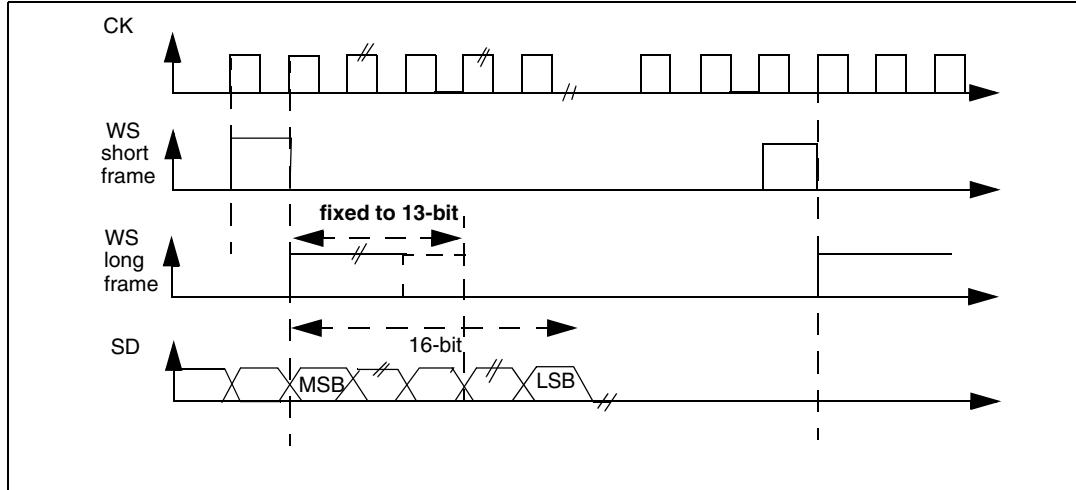
PCM standard

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 SPI_I2SCFGR.

Figure 274. PCM standard waveforms (16-bit)

For long frame synchronization, the WS signal assertion time is fixed 13 bits in master mode.

For short frame synchronization, the WS synchronization signal is only one cycle long.

Figure 275. 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 SPI_I2SCFGR register) even in slave mode.

25.4.4 Clock generator

The I²S bitrate determines the dataflow on the I²S data line and the I²S clock signal frequency.

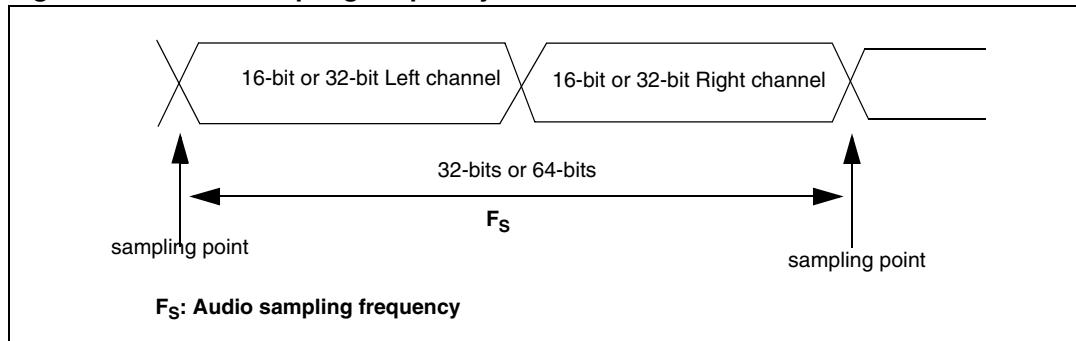
I²S bitrate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I²S bitrate is calculated as follows:

$$\text{I}^2\text{S bitrate} = 16 \times 2 \times F_S$$

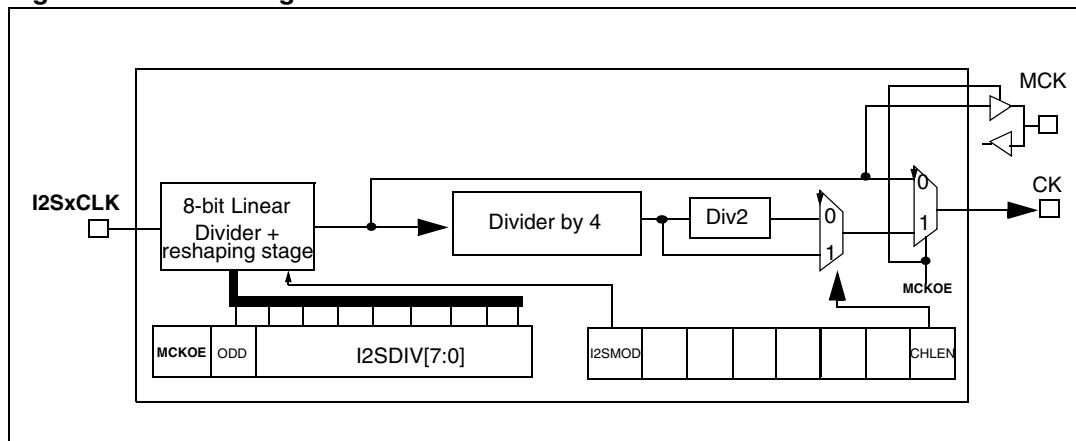
It will be: I^2S bitrate = $32 \times 2 \times F_S$ if the packet length is 32-bit wide.

Figure 276. 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 277. I²S clock generator architecture



1. Where x could be 2 or 3.

Figure 276 presents the communication clock architecture. To achieve high-quality audio performance, the I2SxCLK clock source can be either the PLLI2S output (through R division factor) or an external clock (mapped to I2S_CKIN pin).

The audio sampling frequency can be 192 kHz, 96 kHz, or 48 kHz. In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

When the master clock is generated (MCKOE in the SPI_I2SPR register is set):

$$F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)*8] \text{ when the channel frame is 16-bit wide}$$

$$F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)*4] \text{ when the channel frame is 32-bit wide}$$

When the master clock is disabled (MCKOE bit cleared):

$$F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)] \text{ when the channel frame is 16-bit wide}$$

$$F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)] \text{ when the channel frame is 32-bit wide}$$

Table 100 provides example precision values for different clock configurations.

Note:

Other configurations are possible that allow optimum clock precision.

Table 100. Audio frequency precision (for PLLM VCO = 1 MHz or 2 MHz)⁽¹⁾

Master clock	Target f _S (Hz)	Data format	PLL12SN	PLL12SR	I2SDIV	I2SODD	Real f _S (Hz)	Error
Disabled	8000	16-bit	192	2	187	1	8000	0.0000%
		32-bit	192	3	62	1	8000	0.0000%
	16000	16-bit	192	3	62	1	16000	0.0000%
		32-bit	256	2	62	1	16000	0.0000%
	32000	16-bit	256	2	62	1	32000	0.0000%
		32-bit	256	5	12	1	32000	0.0000%
	48000	16-bit	192	5	12	1	48000	0.0000%
		32-bit	384	5	12	1	48000	0.0000%
	96000	16-bit	384	5	12	1	96000	0.0000%
		32-bit	424	3	11	1	96014.49219	0.0151%
	22050	16-bit	290	3	68	1	22049.87695	0.0006%
		32-bit	302	2	53	1	22050.23438	0.0011%
	44100	16-bit	302	2	53	1	44100.46875	0.0011%
		32-bit	429	4	19	0	44099.50781	0.0011%
Enabled	192000	16-bit	424	3	11	1	192028.9844	0.0151%
		32-bit	258	3	3	1	191964.2813	0.0186%
	8000	don't care	256	5	12	1	8000	0.0000%
	16000	don't care	213	2	13	0	16000.60059	0.0038%
	32000	don't care	213	2	6	1	32001.20117	0.0038%
	48000	don't care	258	3	3	1	47991.07031	0.0186%
	96000	don't care	344	2	3	1	95982.14063	0.0186%
Enabled	22050	don't care	429	4	9	1	22049.75391	0.0011%
	44100	don't care	271	2	6	0	44108.07422	0.0183%

1. This table gives only example values for different clock configurations. Other configurations allowing optimum clock precision are possible.

25.4.5 I²S master mode

The I²S can be configured as follows:

- In master mode for transmission and reception (simplex mode using I2Sx)
- In 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, thanks to the MCKOE bit in the SPI_I2SPR register.

Procedure

1. Select the I2SDIV[7:0] bits in the SPI_I2SPR register to define the serial clock baud rate to reach the proper audio sample frequency. The ODD bit in the SPI_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 SPI_I2SPR register if the master clock MCK needs to be provided to the external DAC/ADC audio component (the I2SDIV and ODD values should be computed depending on the state of the MCK output, for more details refer to [Section 25.4.4: Clock generator](#)).
3. Set the I2SMOD bit in SPI_I2SCFGR to activate the I²S functionalities 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 SPI_I2SCFGR register.
4. If needed, select all the potential interruption sources and the DMA capabilities by writing the SPI_CR2 register.
5. The I2SE bit in SPI_I2SCFGR register must be set.

WS and CK are configured in output mode. MCK is also an output, if the MCKOE bit in SPI_I2SPR is set.

Transmission sequence

The transmission sequence begins when a half-word is written into the Tx buffer.

Assumedly, the first data written into the Tx buffer correspond to the channel Left data. When data are transferred from the Tx buffer to the shift register, TXE is set and data corresponding to the channel Right 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 SPI_CR2 register is set.

For more details about the write operations depending on the I²S standard mode selected, refer to [Section 25.4.3: Supported audio protocols](#).

To ensure a continuous audio data transmission, it is mandatory to write the SPI_DR 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 the transmission mode except for the point 3 (refer to the procedure described in [Section 25.4.5: 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 SPI_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 SPI_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 25.4.3: 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 SPI_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.

25.4.6 I²S slave mode

The I²S can be configured as follows:

- In slave mode for transmission and reception (simplex mode using I2Sx)
- In 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 SPI_I2SCFGR register to reach the I²S functionalities 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 SPI_I2SCFGR register.
2. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPI_CR2 register.
3. The I2SE bit in SPI_I2SCFGR register must be set.

Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the NSS_WS signal requests the transfer of data. The slave has to be enabled before the external master starts the communication. The I²S data register has to be loaded before the master initiates the communication.

For the I²S, MSB justified and LSB justified modes, the first data item to be written into the data register corresponds to the data for the left channel. When the communication starts, the data are transferred from the Tx buffer to the shift register. The TXE flag is then set in order to request the right channel data to be written into the I²S data register.

The CHSIDE flag indicates which channel is to be transmitted. Compared to the master transmission mode, in slave mode, CHSIDE is sensitive to the WS signal coming from the external master. This means that the slave needs to be ready to transmit the first data before the clock is generated by the master. WS assertion corresponds to left channel transmitted first.

Note: *The I2SE has to be written at least two PCLK cycles before the first clock of the master comes on the CK line.*

The data half-word is parallel-loaded into the 16-bit shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI/SD pin MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_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 25.4.3: Supported audio protocols](#).

To secure a continuous audio data transmission, it is mandatory to write the SPI_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 SPI_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 SPI_CR2 register, an interrupt is generated when the UDR flag in the SPI_SR register goes high. In this case, it is mandatory to switch off the I²S and to restart a data transfer starting from the left channel.

To switch off the I²S, by clearing the I2SE bit, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in [Section 25.4.6: I²S slave mode](#)), where the configuration should set the master reception mode using the I2SCFG[1:0] bits in the SPI_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 SPI_SR register is set and an interrupt is generated if the RXNEIE bit is set in the SPI_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 SPI_DR. It is sensitive to the external WS line managed by the external master component.

Clearing the RXNE bit is performed by reading the SPI_DR register.

For more details about the read operations depending the I²S standard mode selected, refer to [Section 25.4.3: Supported audio protocols](#).

If data are received while the precedent 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 SPI_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.*

25.4.7 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 SPI_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 SPI_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 SPI_SR is set and the ERRIE bit in SPI_CR2 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPI_SR status register (once the interrupt source has been cleared).

25.4.8 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 SPI_DR. It is available when the I2SMOD bit in SPI_I2SCFGR is set. An interrupt may be generated if the ERRIE bit in SPI_CR2 is set.

The UDR bit is cleared by a read operation on the SPI_SR register.

Overrun flag (OVR)

This flag is set when data are received and the previous data have not yet been read from SPI_DR. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in SPI_CR2.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPI_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 SPI_DR register followed by a read access to the SPI_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 at a moment when the slave is not expected this

change. If the synchronization is lost, to recover from this state and resynchronize the external master device with the I²S slave device, follow the steps below:

1. Disable the I²S
2. Re-enable it 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 the master and slave device 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.

25.4.9 I²S interrupts

Table 101 provides the list of I²S interrupts.

Table 101. 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	

25.4.10 DMA features

DMA is working in exactly the same way as for the SPI mode. There is no difference on the I²S. Only the CRC feature is not available in I²S mode since there is no data transfer protection system.

25.5 SPI and I²S registers

Refer to for a list of abbreviations used in register descriptions.

25.5.1 SPI control register 1 (SPI_CR1) (not used in I²S mode)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDIMODE:** Bidirectional data mode enable

- 0: 2-line unidirectional data mode selected
- 1: 1-line bidirectional data mode selected

Note: Not used in I²S mode

Bit 14 **BIDIOE:** Output enable in bidirectional mode

- This bit combined with the BIDI mode bit selects the direction of transfer in bidirectional mode
- 0: Output disabled (receive-only mode)
- 1: Output enabled (transmit-only mode)

Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used.

Not used in I²S mode

Bit 13 **CRCEN:** Hardware CRC calculation enable

- 0: CRC calculation disabled
- 1: CRC calculation enabled

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation

Not used in I²S mode

Bit 12 **CRCNEXT:** CRC transfer next

- 0: Data phase (no CRC phase)
- 1: Next transfer is CRC (CRC phase)

Note: When the SPI is configured in full duplex or transmitter only modes, CRCNEXT must be written as soon as the last data is written to the SPI_DR register.

When the SPI is configured in receiver only mode, CRCNEXT must be set after the second last data reception.

This bit should be kept cleared when the transfers are managed by DMA.

Not used in I²S mode

Bit 11 **DFF:** Data frame format

- 0: 8-bit data frame format is selected for transmission/reception
- 1: 16-bit data frame format is selected for transmission/reception

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation

Not used in I²S mode

Bit 10 **RXONLY:** Receive only

This bit combined with the BIDI mode bit selects the direction of transfer in 2-line unidirectional mode. This bit is also useful in a multislave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.

- 0: Full duplex (Transmit and receive)
- 1: Output disabled (Receive-only mode)

Note: 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: 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 IO value of the NSS pin is ignored.

Note: Not used in I²S mode and SPI TI mode

Bit 7 **LSBFIRST:** Frame format

- 0: MSB transmitted first
- 1: LSB transmitted first

Note: This bit should not be changed when communication is ongoing.

Not used in I²S mode and SPI TI mode

Bit 6 **SPE:** SPI enable

- 0: Peripheral disabled
- 1: Peripheral enabled

Note: 1- Not used in I²S mode.

Note: 2- When disabling the SPI, follow the procedure described in [Section 25.3.8: Disabling the SPI](#).

Bits 5:3 **BR[2:0]:** Baud rate control

000: f _{PCLK} /2	100: f _{PCLK} /32
001: f _{PCLK} /4	101: f _{PCLK} /64
010: f _{PCLK} /8	110: f _{PCLK} /128
011: f _{PCLK} /16	111: f _{PCLK} /256

Note: These bits should not be changed when communication is ongoing.

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.

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.

Not used in I²S mode and SPI TI mode

Bit 0 **CPHA:** Clock phase

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

Note: This bit should not be changed when communication is ongoing.

Note: Not used in I²S mode and SPI TI mode

25.5.2 SPI control register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TXEIE	RXNEIE	ERRIE	FRF		SSOE	TXDMAEN	RXDMAEN
Reserved								rw	rw	rw	rw	Res.	rw	rw	rw

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

Bit 7 **TXEIE:** Tx buffer empty interrupt enable

0: TXE interrupt masked

1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE:** RX buffer not empty interrupt enable

0: RXNE interrupt masked

1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE:** Error interrupt enable

This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode and UDR, OVR in I²S 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: Not used in I²S mode

Bit 3 Reserved. Forced to 0 by hardware.

Bit 2 **SSOE:** SS output enable

0: SS output is disabled in master mode and the cell can work in multimaster configuration

1: SS output is enabled in master mode and when the cell is enabled. The cell cannot work in a multimaster environment.

Note: Not used in I²S mode and SPI TI mode

Bit 1 **TXDMAEN:** Tx buffer DMA enable

When this bit is set, the DMA request is made whenever the TXE flag is set.

0: Tx buffer DMA disabled

1: Tx buffer DMA enabled

Bit 0 **RXDMAEN:** Rx buffer DMA enable

When this bit is set, the DMA request is made whenever the RXNE flag is set.

0: Rx buffer DMA disabled

1: Rx buffer DMA enabled

25.5.3 SPI status register (SPI_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TIFRFE	BSY	OVR	MODF	CRC ERR	UDR	CHSID E	TXE	RXNE	
r	r	r	r	r	rc_w0	r	r	r							r

Bits 15:9 Reserved. Forced to 0 by hardware.

Bit 8 **TIFRFE**: TI frame format error

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: *BSY flag must be used with caution: refer to [Section 25.3.7: Status flags](#) and [Section 25.3.8: Disabling the SPI](#).*

Bit 6 **OVR**: Overrun flag

0: No overrun occurred

1: Overrun occurred

This flag is set by hardware and reset by a software sequence. Refer to [Section 25.4.8 on page 699](#) 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 25.3.10 on page 682](#) for the software sequence.

Note: *Not used in I²S mode*

Bit 4 **CRCERR**: CRC error flag

0: CRC value received matches the SPI_RXCRCR value

1: CRC value received does not match the SPI_RXCRCR value

This flag is set by hardware and cleared by software writing 0.

Note: *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 [Section 25.4.8 on page 699](#) for the software sequence.

Note: *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: *Not used for the SPI mode. No meaning in PCM mode*

Bit 1 **TXE**: Transmit buffer empty

0: Tx buffer not empty

1: Tx buffer empty

Bit 0 **RXNE**: Receive buffer not empty
 0: Rx buffer empty
 1: Rx buffer not empty

25.5.4 SPI data register (SPI_DR)

Address offset: 0x0C

Reset value: 0x0000

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

Bits 15:0 **DR[15:0]**: Data register

Data received or to be transmitted.

The data register is split into 2 buffers - one for writing (Transmit Buffer) and another one for reading (Receive buffer). A write to the data register will write into the Tx buffer and a read from the data register will return the value held in the Rx buffer.

Notes for the SPI mode:

Depending on the data frame format selection bit (DFF in SPI_CR1 register), the data sent or received is either 8-bit or 16-bit. This selection has to be made before enabling the SPI to ensure correct operation.

For an 8-bit data frame, the buffers are 8-bit and only the LSB of the register (SPI_DR[7:0]) is used for transmission/reception. When in reception mode, the MSB of the register (SPI_DR[15:8]) is forced to 0.

For a 16-bit data frame, the buffers are 16-bit and the entire register, SPI_DR[15:0] is used for transmission/reception.

25.5.5 SPI CRC polynomial register (SPI_CRCPR) (not used in I²S mode)

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: Not used for the I²S mode.

25.5.6 SPI RX CRC register (SPI_RXCRCR) (not used in I²S mode)

Address offset: 0x14

Reset value: 0x0000

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

Bits 15:0 **RXCRC[15:0]:** Rx CRC register

When CRC calculation is enabled, the RxCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPI_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (DFF bit of SPI_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (DFF bit of the SPI_CR1 register is set). CRC calculation is done based on any CRC16 standard.

*Note: A read to this register when the BSY Flag is set could return an incorrect value.
Not used for the I²S mode.*

25.5.7 SPI TX CRC register (SPI_TXCRCR) (not used in I²S mode)

Address offset: 0x18

Reset value: 0x0000

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

Bits 15:0 **TXCRC[15:0]:** Tx CRC register

When CRC calculation is enabled, the TxCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPI_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (DFF bit of SPI_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (DFF bit of the SPI_CR1 register is set). CRC calculation is done based on any CRC16 standard.

*Note: A read to this register when the BSY flag is set could return an incorrect value.
Not used for I²S mode.*

25.5.8 SPI_I²S configuration register (SPI_I2SCFGR)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				I2SMOD	I2SE	I2SCFG		PCMSYNC		I2SSTD		CKPOL	DATLEN		CHLEN
Reserved				rw	rw	rw	rw	rw	Reserved	rw	rw	rw	rw	rw	rw

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

Bit 11 **I2SMOD**: I2S mode selection

- 0: SPI mode is selected
- 1: I2S mode is selected

Note: This bit should be configured when the SPI or I²S is disabled

Bit 10 **I2SE**: I2S Enable

- 0: I²S peripheral is disabled
- 1: I²S peripheral is enabled

Note: Not used in SPI mode

Bit 9:8 **I2SCFG**: I2S configuration mode

- 00: Slave - transmit
- 01: Slave - receive
- 10: Master - transmit
- 11: Master - receive

Note: This bit should be configured when the I²S is disabled.

Not used for the 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)

Not used for the SPI mode

Bit 6 Reserved: forced at 0 by hardware

Bit 5:4 **I2SSTD**: I2S standard selection

- 00: I²S Phillips 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 25.4.3 on page 686](#). *Not used in SPI mode*.

Note: For correct operation, these bits should be configured when the I²S is disabled.

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.

Not used in SPI mode

Bit 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.
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. Not used in SPI mode.

Note: For correct operation, this bit should be configured when the I²S is disabled.

25.5.9 SPI_I²S prescaler register (SPI_I2SPR)

Address offset: 0x20

Reset value: 0000 0010 (0x0002)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MCKOE	ODD	I2SDIV					
		rw		rw		rw									

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

Bit 9 **MCKOE**: Master clock output enable

- 0: Master clock output is disabled
- 1: Master clock output is enabled

Note: This bit should be configured when the I²S is disabled. It is used only when the I²S is in master mode.

Not used in SPI mode.

Bit 8 **ODD**: Odd factor for the prescaler

- 0: real divider value is = I2SDIV *2
- 1: real divider value is = (I2SDIV * 2)+1

Refer to [Section 25.4.4 on page 692](#). Not used in SPI mode.

Note: This bit should be configured when the I²S is disabled. It is used only when the I²S is in master mode.

Bit 7:0 **I2SDIV**: I2S Linear prescaler

I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values.

Refer to [Section 25.4.4 on page 692](#). Not used in SPI mode.

Note: These bits should be configured when the I²S is disabled. It is used only when the I²S is in master mode.

25.5.10 SPI register map

The table provides shows the SPI register map and reset values.

Table 102. SPI register map and reset values

Refer to [Table 1 on page 50](#) for the register boundary addresses.

26 Secure digital input/output interface (SDIO)

26.1 SDIO main features

The SD/SDIO MMC card host interface (SDIO) provides an interface between the APB2 peripheral bus and MultiMediaCards (MMCs), SD memory cards, SDIO cards and CE-ATA devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.mmca.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcards.org.

CE-ATA system specifications are available through the CE-ATA workgroup website at www.ce-ata.org.

The SDIO features include the following:

- Full compliance with *MultiMediaCard System Specification Version 4.2*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit
- Full compatibility with previous versions of MultiMediaCards (forward compatibility)
- Full compliance with *SD Memory Card Specifications Version 2.0*
- Full compliance with *SD I/O Card Specification Version 2.0*: card support for two different databus modes: 1-bit (default) and 4-bit
- Full support of the CE-ATA features (full compliance with *CE-ATA digital protocol Rev1.1*)
- Data transfer up to 48 MHz for the 8 bit mode
- Data and command output enable signals to control external bidirectional drivers.

- Note: 1 *The SDIO does not have an SPI-compatible communication mode.*
- 2 *The SD memory card protocol is a superset of the MultiMediaCard protocol as defined in the MultiMediaCard system specification V2.11. Several commands required for SD memory devices are not supported by either SD I/O-only cards or the I/O portion of combo cards. Some of these commands have no use in SD I/O devices, such as erase commands, and thus are not supported in the SDIO. In addition, several commands are different between SD memory cards and SD I/O cards and thus are not supported in the SDIO. For details refer to SD I/O card Specification Version 1.0. CE-ATA is supported over the MMC electrical interface using a protocol that utilizes the existing MMC access primitives. The interface electrical and signaling definition is as defined in the MMC reference.*

The MultiMediaCard/SD bus connects cards to the controller.

The current version of the SDIO supports only one SD/SDIO/MMC4.2 card at any one time and a stack of MMC4.1 or previous.

26.2 SDIO bus topology

Communication over the bus is based on command and data transfers.

The basic transaction on the MultiMediaCard/SD/SD I/O bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers to/from SD/SDIO memory cards are done in data blocks. Data transfers to/from MMC are done data blocks or streams. Data transfers to/from the CE-ATA Devices are done in data blocks.

Figure 278. SDIO “no response” and “no data” operations

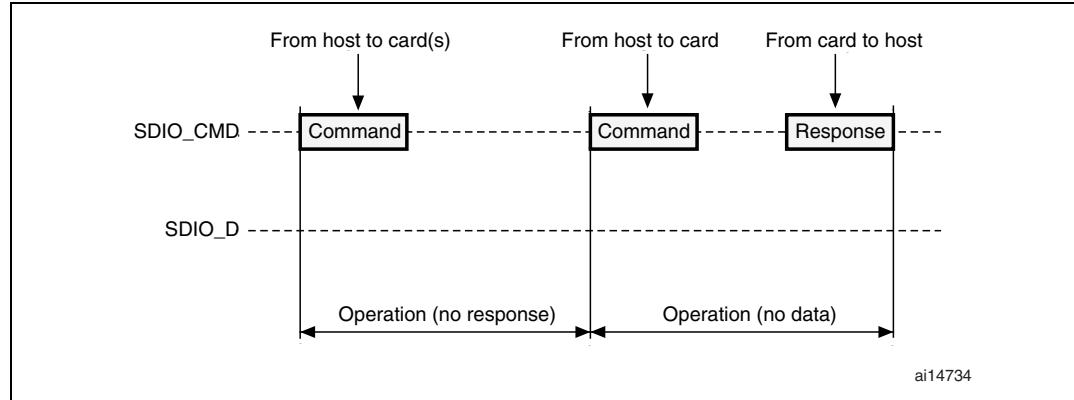


Figure 279. SDIO (multiple) block read operation

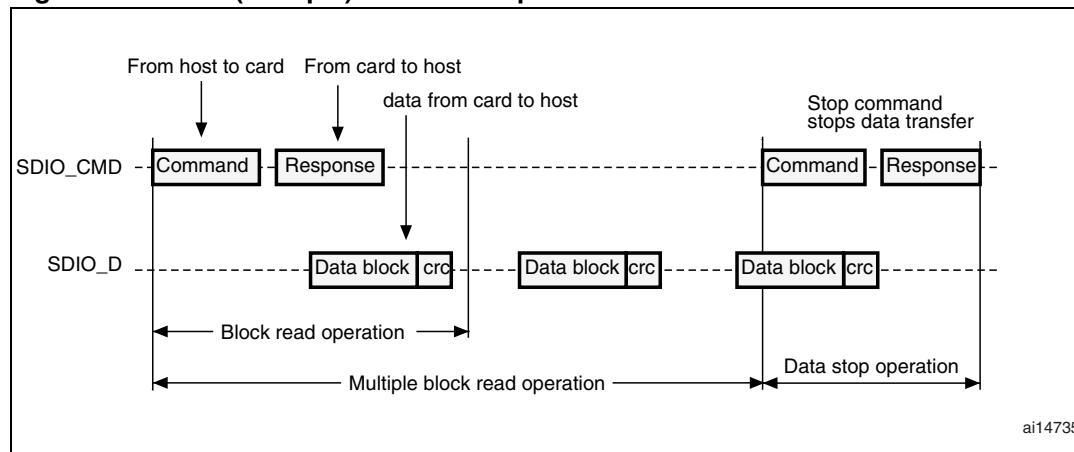
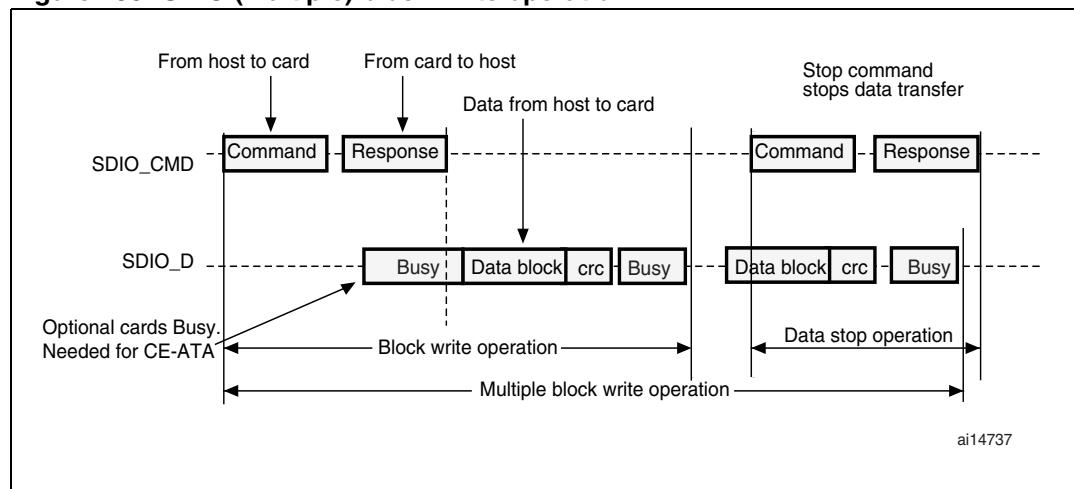


Figure 280. SDIO (multiple) block write operation



Note: The SDIO will not send any data as long as the Busy signal is asserted (SDIO_D0 pulled low).

Figure 281. SDIO sequential read operation

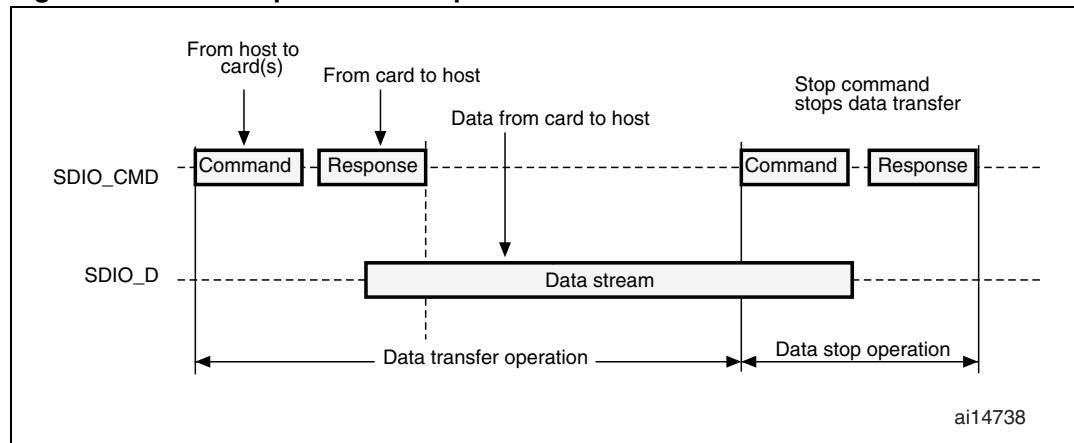
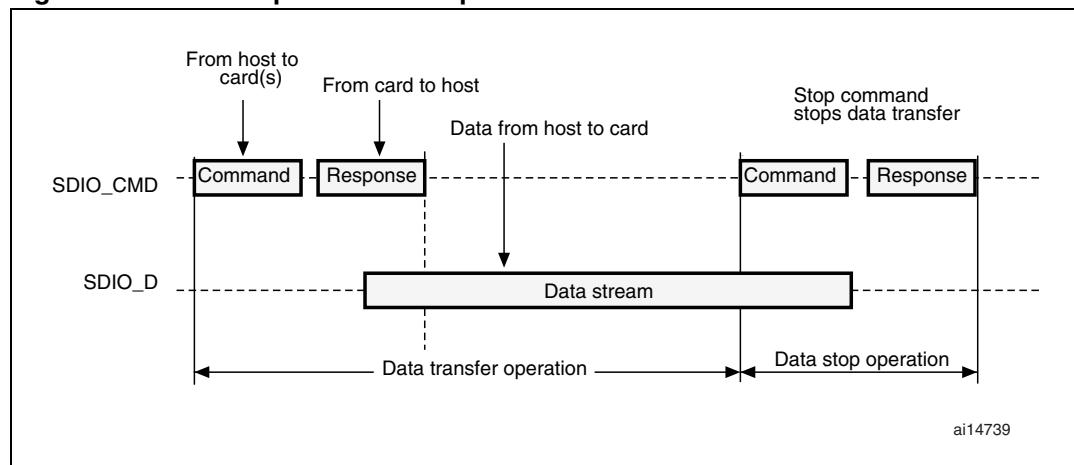


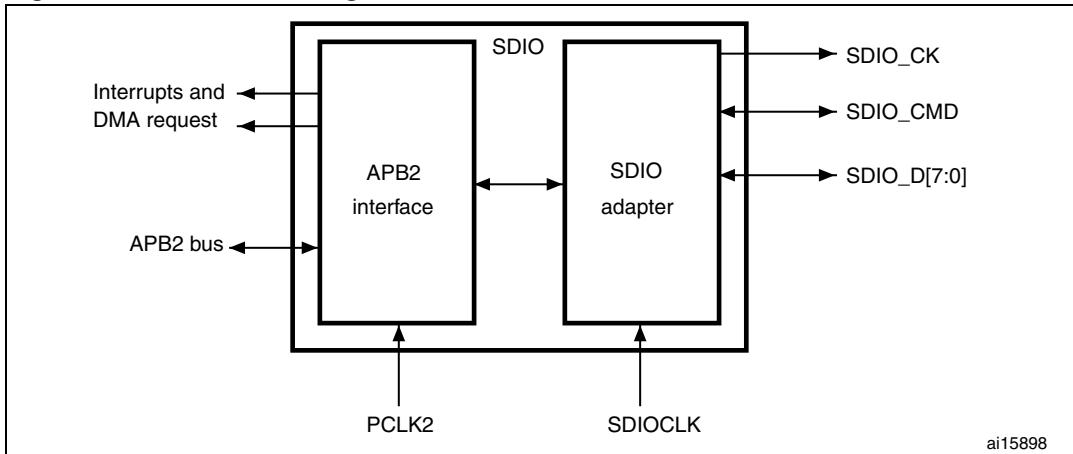
Figure 282. SDIO sequential write operation



26.3 SDIO functional description

The SDIO consists of two parts:

- The SDIO adapter block provides all functions specific to the MMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The APB2 interface accesses the SDIO adapter registers, and generates interrupt and DMA request signals.

Figure 283. SDIO block diagram

By default SDIO_D0 is used for data transfer. After initialization, the host can change the databus width.

If a MultiMediaCard is connected to the bus, SDIO_D0, SDIO_D[3:0] or SDIO_D[7:0] can be used for data transfer. MMC V3.31 or previous, supports only 1 bit of data so only SDIO_D0 can be used.

If an SD or SD I/O card is connected to the bus, data transfer can be configured by the host to use SDIO_D0 or SDIO_D[3:0]. All data lines are operating in push-pull mode.

SDIO_CMD has two operational modes:

- Open-drain for initialization (only for MMCV3.31 or previous)
- Push-pull for command transfer (SD/SD I/O card MMC4.2 use push-pull drivers also for initialization)

SDIO_CK is the clock to the card: one bit is transferred on both command and data lines with each clock cycle. The clock frequency can vary between 0 MHz and 20 MHz (for a MultiMediaCard V3.31), between 0 and 48 MHz for a MultiMediaCard V4.0/4.2, or between 0 and 25 MHz (for an SD/SD I/O card).

The SDIO uses two clock signals:

- SDIO adapter clock (SDIOCLK = 48 MHz)
- APB2 bus clock (PCLK2)

PCLK2 and SDIO_CK clock frequencies must respect the following condition:

$$\text{Frequenc}(PCLK2) \geq 3/8 \times \text{Frequency}(SDIO_CK)$$

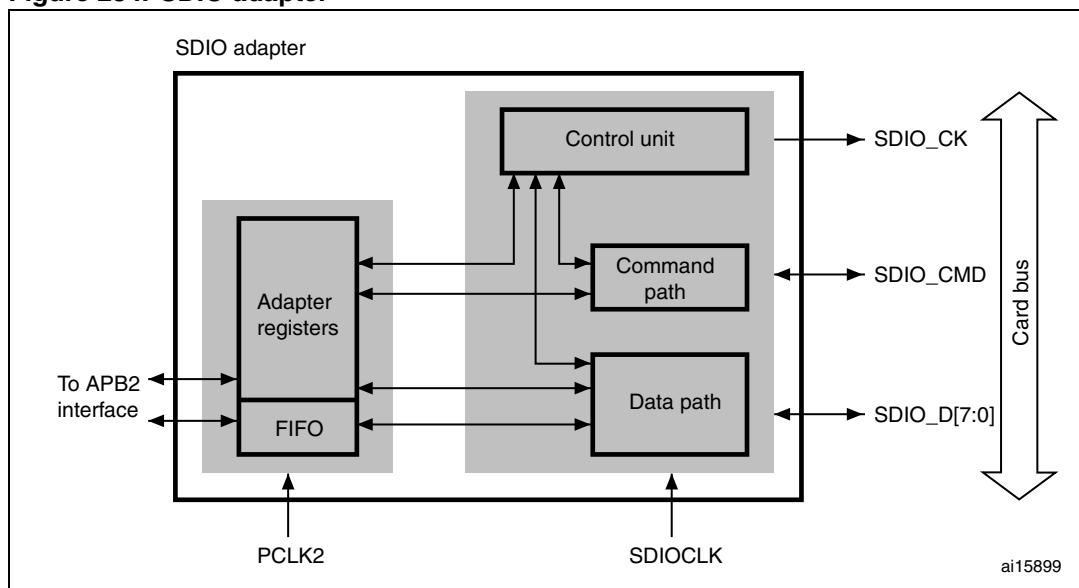
The signals shown in [Table 103](#) are used on the MultiMediaCard/SD/SD I/O card bus.

Table 103. SDIO I/O definitions

Pin	Direction	Description
SDIO_CK	Output	MultiMediaCard/SD/SDIO card clock. This pin is the clock from host to card.
SDIO_CMD	Bidirectional	MultiMediaCard/SD/SDIO card command. This pin is the bidirectional command/response signal.
SDIO_D[7:0]	Bidirectional	MultiMediaCard/SD/SDIO card data. These pins are the bidirectional databus.

26.3.1 SDIO adapter

Figure 284 shows a simplified block diagram of an SDIO adapter.

Figure 284. SDIO adapter

The SDIO adapter is a multimedia/secure digital memory card bus master that provides an interface to a multimedia card stack or to a secure digital memory card. It consists of five subunits:

- Adapter register block
- Control unit
- Command path
- Data path
- Data FIFO

Note:

The adapter registers and FIFO use the APB2 bus clock domain (PCLK2). The control unit, command path and data path use the SDIO adapter clock domain (SDIOCLK).

Adapter register block

The adapter register block contains all system registers. This block also generates the signals that clear the static flags in the multimedia card. The clear signals are generated when 1 is written into the corresponding bit location in the SDIO Clear register.

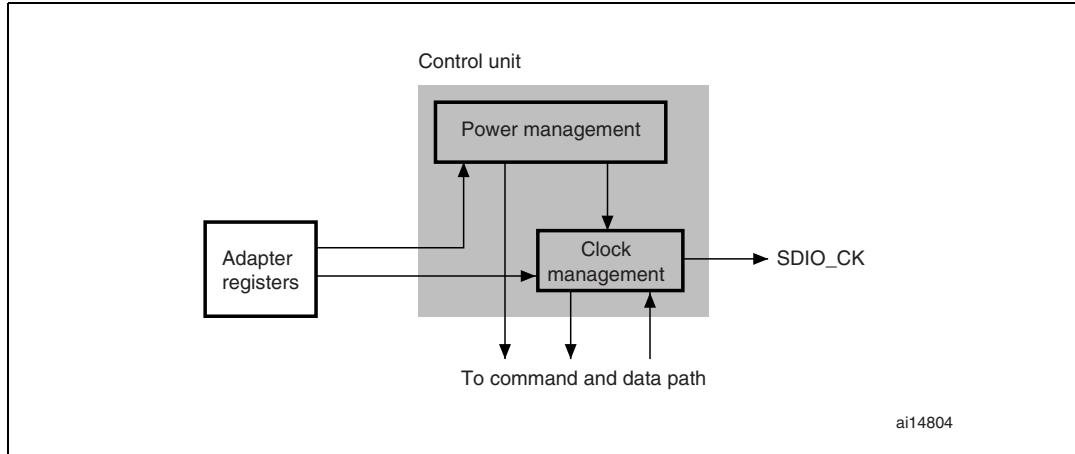
Control unit

The control unit contains the power management functions and the clock divider for the memory card clock.

There are three power phases:

- power-off
- power-up
- power-on

Figure 285. Control unit



The control unit is illustrated in [Figure 285](#). It consists of a power management subunit and a clock management subunit.

The power management subunit disables the card bus output signals during the power-off and power-up phases.

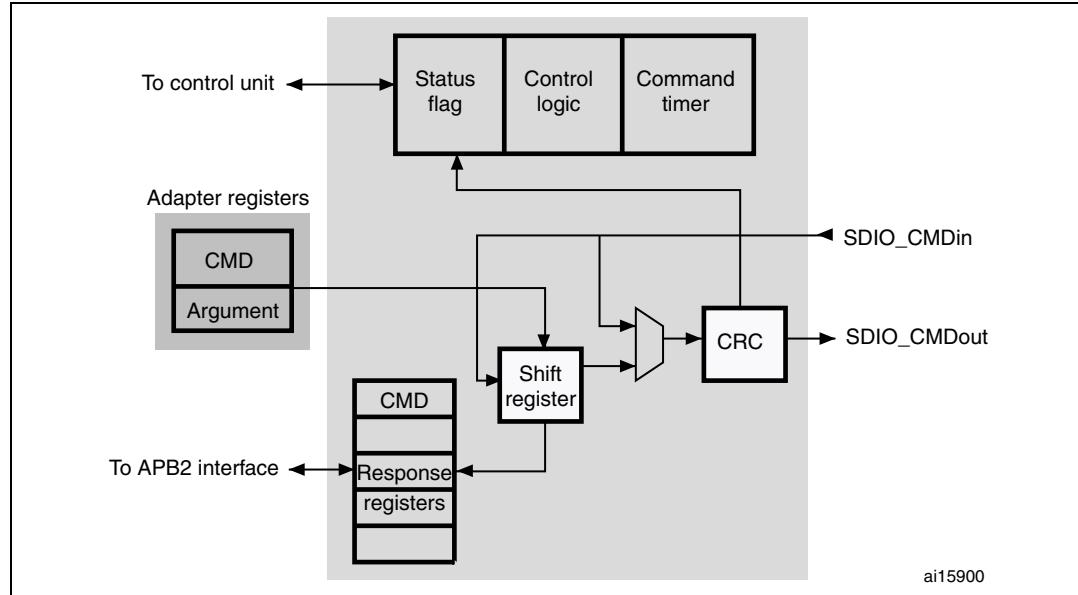
The clock management subunit generates and controls the SDIO_CK signal. The SDIO_CK output can use either the clock divide or the clock bypass mode. The clock output is inactive:

- after reset
- during the power-off or power-up phases
- if the power saving mode is enabled and the card bus is in the Idle state (eight clock periods after both the command and data path subunits enter the Idle phase)

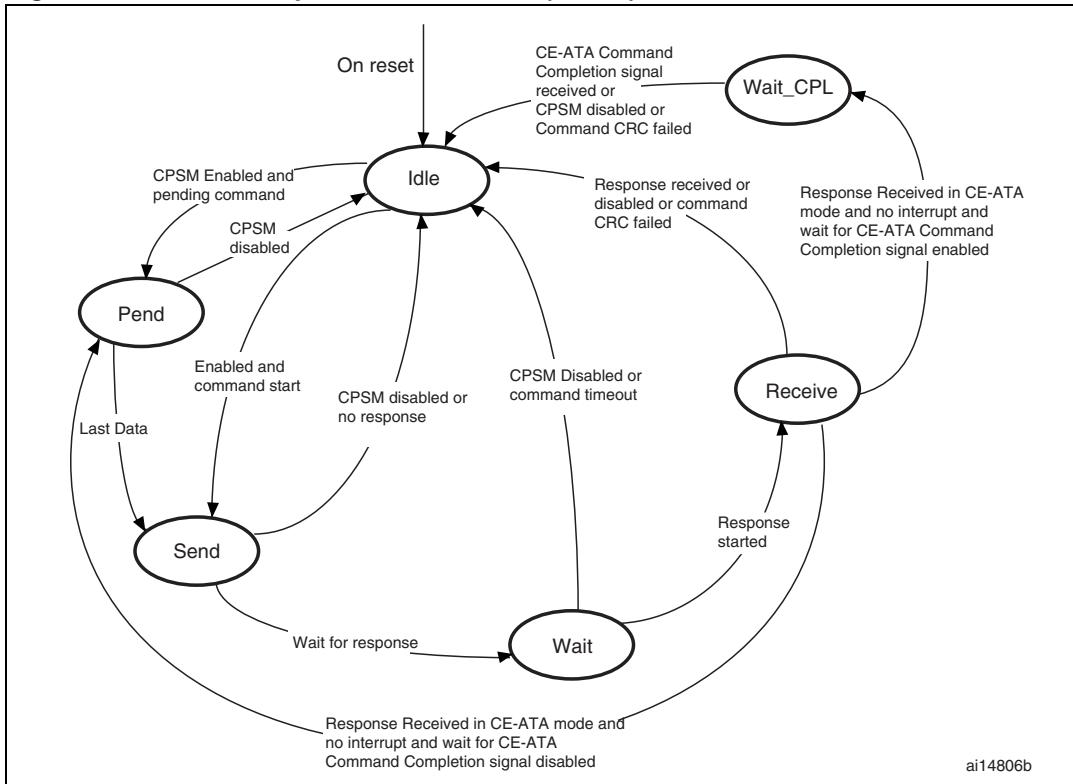
Command path

The command path unit sends commands to and receives responses from the cards.

Figure 286. SDIO adapter command path



- Command path state machine (CPSM)
 - When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent, the command path state machine (CPSM) sets the status flags and enters the Idle state if a response is not required. If a response is required, it waits for the response (see [Figure 287 on page 717](#)). When the response is received, the received CRC code and the internally generated code are compared, and the appropriate status flags are set.

Figure 287. Command path state machine (CPSM)

ai14806b

When the Wait state is entered, the command timer starts running. If the timeout is reached before the CPSM moves to the Receive state, the timeout flag is set and the Idle state is entered.

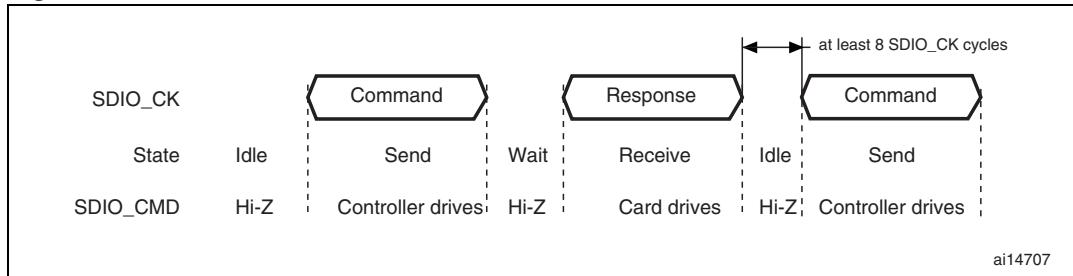
Note:

The command timeout has a fixed value of 64 SDIO_CK clock periods.

If the interrupt bit is set in the command register, the timer is disabled and the CPSM waits for an interrupt request from one of the cards. If a pending bit is set in the command register, the CPSM enters the Pend state, and waits for a CmdPend signal from the data path subunit. When CmdPend is detected, the CPSM moves to the Send state. This enables the data counter to trigger the stop command transmission.

Note:

The CPSM remains in the Idle state for at least eight SDIO_CK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the card response.

Figure 288. SDIO command transfer

- **Command format**

- **Command:** a command is a token that starts an operation. Commands are sent from the host either to a single card (addressed command) or to all connected cards (broadcast command are available for MMC V3.31 or previous). Commands are transferred serially on the CMD line. All commands have a fixed length of 48 bits. The general format for a command token for MultiMediaCards, SD-Memory cards and SDIO-Cards is shown in [Table 104](#). CE-ATA commands are an extension of MMC commands V4.2, and so have the same format.

The command path operates in a half-duplex mode, so that commands and responses can either be sent or received. If the CPSM is not in the Send state, the SDIO_CMD output is in the Hi-Z state, as shown in [Figure 288 on page 718](#). Data on SDIO_CMD are synchronous with the rising edge of SDIO_CK. [Table](#) shows the command format.

Table 104. Command format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	-	Command index
[39:8]	32	-	Argument
[7:1]	7	-	CRC7
0	1	1	End bit

- **Response:** a response is a token that is sent from an addressed card (or synchronously from all connected cards for MMC V3.31 or previous), to the host as an answer to a previously received command. Responses are transferred serially on the CMD line.

The SDIO supports two response types. Both use CRC error checking:

- 48 bit short response
- 136 bit long response

Note: *If the response does not contain a CRC (CMD1 response), the device driver must ignore the CRC failed status.*

Table 105. Short response format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	-	Command index
[39:8]	32	-	Argument
[7:1]	7	-	CRC7(or 1111111)
0	1	1	End bit

Table 106. Long response format

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127	-	CID or CSD (including internal CRC7)
0	1	1	End bit

The command register contains the command index (six bits sent to a card) and the command type. These determine whether the command requires a response, and whether the response is 48 or 136 bits long (see [Section 26.9.4 on page 753](#)). The command path implements the status flags shown in [Table 107](#):

Table 107. Command path status flags

Flag	Description
CMDREND	Set if response CRC is OK.
CCRCFAIL	Set if response CRC fails.
CMDSENT	Set when command (that does not require response) is sent
CTIMEOUT	Response timeout.
CMDACT	Command transfer in progress.

The CRC generator calculates the CRC checksum for all bits before the CRC code. This includes the start bit, transmitter bit, command index, and command argument (or card status). The CRC checksum is calculated for the first 120 bits of CID or CSD for the long response format. Note that the start bit, transmitter bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

$$\text{CRC}[6:0] = \text{Remainder } [(M(x) * x^7) / G(x)]$$

$$G(x) = x^7 + x^3 + 1$$

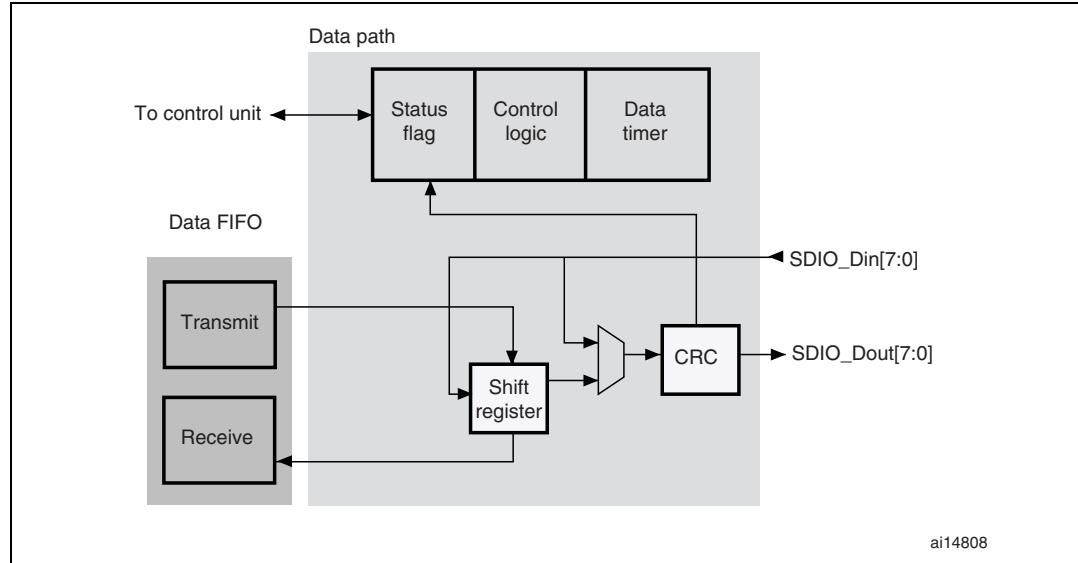
$$M(x) = (\text{start bit}) * x^{39} + \dots + (\text{last bit before CRC}) * x^0, \text{ or}$$

$$M(x) = (\text{start bit}) * x^{119} + \dots + (\text{last bit before CRC}) * x^0$$

Data path

The data path subunit transfers data to and from cards. [Figure 289](#) shows a block diagram of the data path.

Figure 289. Data path



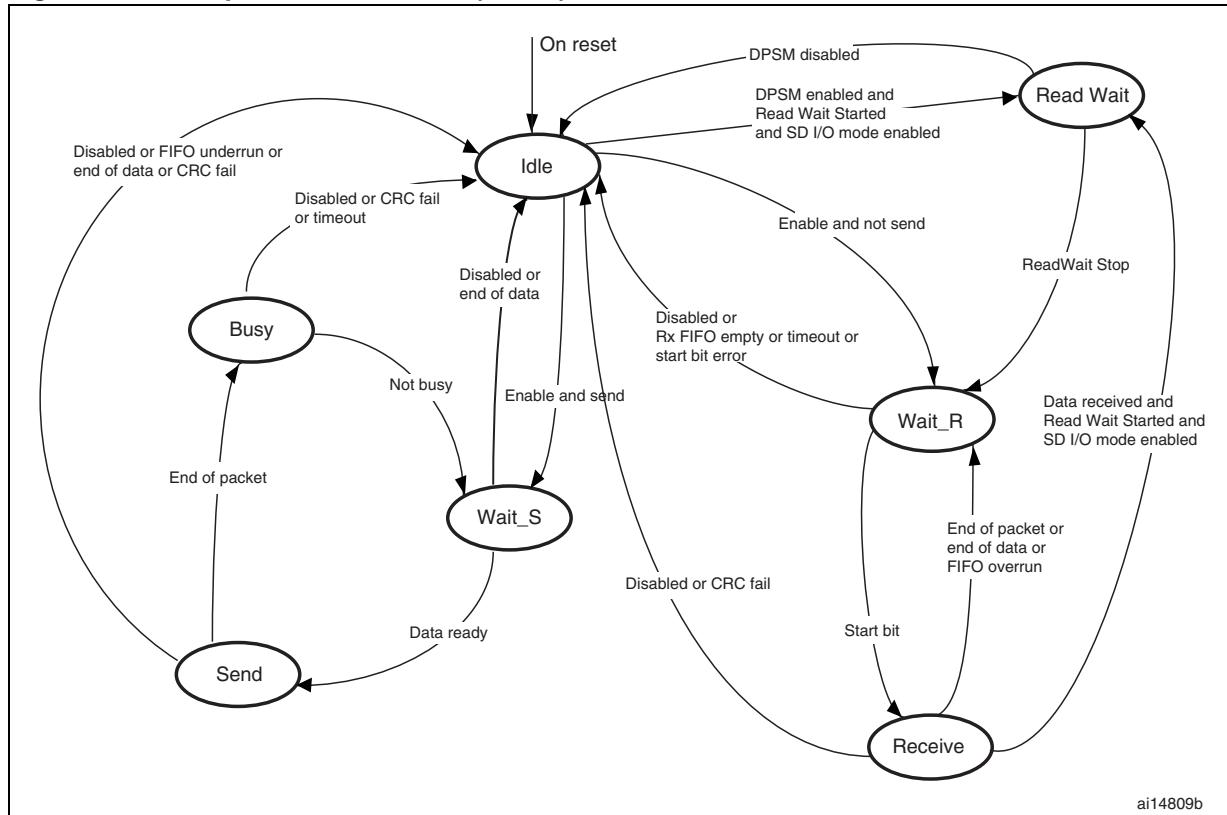
The card databus width can be programmed using the clock control register. If the 4-bit wide bus mode is enabled, data is transferred at four bits per clock cycle over all four data signals (SDIO_D[3:0]). If the 8-bit wide bus mode is enabled, data is transferred at eight bits per clock cycle over all eight data signals (SDIO_D[7:0]). If the wide bus mode is not enabled, only one bit per clock cycle is transferred over SDIO_D0.

Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

Data path state machine (DPSM)

The DPSM operates at SDIO_CK frequency. Data on the card bus signals is synchronous to the rising edge of SDIO_CK. The DPSM has six states, as shown in [Figure 290: Data path state machine \(DPSM\)](#).

Figure 290. Data path state machine (DPSM)

- **Idle:** the data path is inactive, and the SDIO_D[7:0] outputs are in Hi-Z. When the data control register is written and the enable bit is set, the DPSM loads the data counter with a new value and, depending on the data direction bit, moves to either the Wait_S or the Wait_R state.
- **Wait_R:** if the data counter equals zero, the DPSM moves to the Idle state when the receive FIFO is empty. If the data counter is not zero, the DPSM waits for a start bit on SDIO_D. The DPSM moves to the Receive state if it receives a start bit before a timeout, and loads the data block counter. If it reaches a timeout before it detects a start bit, or a start bit error occurs, it moves to the Idle state and sets the timeout status flag.
- **Receive:** serial data received from a card is packed in bytes and written to the data FIFO. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
 - In block mode, when the data block counter reaches zero, the DPSM waits until it receives the CRC code. If the received code matches the internally generated CRC code, the DPSM moves to the Wait_R state. If not, the CRC fail status flag is set and the DPSM moves to the Idle state.
 - In stream mode, the DPSM receives data while the data counter is not zero. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.
- If a FIFO overrun error occurs, the DPSM sets the FIFO error flag and moves to the Idle state:
- **Wait_S:** the DPSM moves to the Idle state if the data counter is zero. If not, it waits until the data FIFO empty flag is deasserted, and moves to the Send state.

Note: The DPSM remains in the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements, where N_{WR} is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- Send: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
 - In block mode, when the data block counter reaches zero, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state.
 - In stream mode, the DPSM sends data to a card while the enable bit is high and the data counter is not zero. It then moves to the Idle state.
 If a FIFO underrun error occurs, the DPSM sets the FIFO error flag and moves to the Idle state.
- Busy: the DPSM waits for the CRC status flag:
 - If it does not receive a positive CRC status, it moves to the Idle state and sets the CRC fail status flag.
 - If it receives a positive CRC status, it moves to the Wait_S state if SDIO_D0 is not low (the card is not busy).
 If a timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag and moves to the Idle state.
- Data: data can be transferred from the card to the host or vice versa. Data is transferred via the data lines. They are stored in a FIFO of 32 words, each word is 32 bits wide.

Table 108. Data token format

Description	Start bit	Data	CRC16	End bit
Block Data	0	-	yes	1
Stream Data	0	-	no	1

Data FIFO

The data FIFO (first-in-first-out) subunit is a data buffer with a transmit and receive unit.

The FIFO contains a 32-bit wide, 32-word deep data buffer, and transmit and receive logic. Because the data FIFO operates in the APB2 clock domain (PCLK2), all signals from the subunits in the SDIO clock domain (SDIOCLK) are resynchronized.

Depending on the TXACT and RXACT flags, the FIFO can be disabled, transmit enabled, or receive enabled. TXACT and RXACT are driven by the data path subunit and are mutually exclusive:

- The transmit FIFO refers to the transmit logic and data buffer when TXACT is asserted
- The receive FIFO refers to the receive logic and data buffer when RXACT is asserted
- Transmit FIFO:
Data can be written to the transmit FIFO through the APB2 interface when the SDIO is enabled for transmission.
The transmit FIFO is accessible via 32 sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out.
If the transmit FIFO is disabled, all status flags are deasserted. The data path subunit asserts TXACT when it transmits data.

Table 109. Transmit FIFO status flags

Flag	Description
TXFIFOF	Set to high when all 32 transmit FIFO words contain valid data.
TXFIFOE	Set to high when the transmit FIFO does not contain valid data.
TXFIFOHE	Set to high when 8 or more transmit FIFO words are empty. This flag can be used as a DMA request.
TXDAVL	Set to high when the transmit FIFO contains valid data. This flag is the inverse of the TXFIFOE flag.
TXUNDERR	Set to high when an underrun error occurs. This flag is cleared by writing to the SDIO Clear register.

- Receive FIFO

When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. If the receive FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. The data path subunit asserts RXACT when it receives data. [Table 110](#) lists the receive FIFO status flags. The receive FIFO is accessible via 32 sequential addresses.

Table 110. Receive FIFO status flags

Flag	Description
RXFIFOF	Set to high when all 32 receive FIFO words contain valid data
RXFIFOE	Set to high when the receive FIFO does not contain valid data.
RXFIFOHF	Set to high when 8 or more receive FIFO words contain valid data. This flag can be used as a DMA request.
RXDAVL	Set to high when the receive FIFO is not empty. This flag is the inverse of the RXFIFOE flag.
RXOVERR	Set to high when an overrun error occurs. This flag is cleared by writing to the SDIO Clear register.

26.3.2 SDIO APB2 interface

The APB2 interface generates the interrupt and DMA requests, and accesses the SDIO adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt/DMA logic.

SDIO interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the selected status flags is high. A mask register is provided to allow selection of the conditions that will generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set.

SDIO/DMA interface - procedure for data transfers between the SDIO and memory

In the example shown, the transfer is from the SDIO host controller to an MMC (512 bytes using CMD24 (WRITE_BLOCK)). The SDIO FIFO is filled by data stored in a memory using the DMA controller.

1. Do the card identification process
2. Increase the SDIO_CK frequency
3. Select the card by sending CMD7
4. Configure the DMA2 as follows:
 - a) Enable DMA2 controller and clear any pending interrupts.
 - b) Program the DMA2_Stream3 or DMA2_Stream6 Channel4 source address register with the memory location's base address and DMA2_Stream3 or DMA2_Stream6 Channel4 destination address register with the SDIO_FIFO register address.
 - c) Program DMA2_Stream3 or DMA2_Stream6 Channel4 control register (memory increment, not peripheral increment, peripheral and source width is word size).
 - d) Program DMA2_Stream3 or DMA2_Stream6 Channel4 to select the peripheral as flow controller (set PFCTRL bit in DMA_S3CR or DMA_S6CR configuration register).
 - e) Configure the incremental burst transfer to 4 beats (at least from peripheral side) in DMA2_Stream3 or DMA2_Stream6 Channel4.
 - f) Enable DMA2_Stream3 or DMA2_Stream6 Channel4

5. Send CMD24 (WRITE_BLOCK) as follows:
 - a) Program the SDIO data length register (SDIO data timer register should be already programmed before the card identification process).
 - b) Program the SDIO argument register with the address location of the card where data is to be transferred.
 - c) Program the SDIO command register: CmdIndex with 24 (WRITE_BLOCK); WaitResp with '1' (SDIO card host waits for a response); CPSMEN with '1' (SDIO card host enabled to send a command). Other fields are at their reset value.
 - d) Wait for SDIO_STA[6] = CMDREND interrupt, then program the SDIO data control register: DTEN with '1' (SDIO card host enabled to send data); DTDIR with '0' (from controller to card); DTMODE with '0' (block data transfer); DMAEN with '1' (DMA enabled); DBLOCKSIZE with 0x9 (512 bytes). Other fields are don't care.
 - e) Wait for SDIO_STA[10] = DBCKEND.
6. Check that no channels are still enabled by polling the DMA Enabled Channel Status register.

26.4 Card functional description

26.4.1 Card identification mode

While in card identification mode the host resets all cards, validates the operation voltage range, identifies cards and sets a relative card address (RCA) for each card on the bus. All data communications in the card identification mode use the command line (CMD) only.

26.4.2 Card reset

The GO_IDLE_STATE command (CMD0) is the software reset command and it puts the MultiMediaCard and SD memory in the Idle state. The IO_RW_DIRECT command (CMD52) resets the SD I/O card. After power-up or CMD0, all cards output bus drivers are in the high-impedance state and the cards are initialized with a default relative card address (RCA=0x0001) and with a default driver stage register setting (lowest speed, highest driving current capability).

26.4.3 Operating voltage range validation

All cards can communicate with the SDIO card host using any operating voltage within the specification range. The supported minimum and maximum V_{DD} values are defined in the operation conditions register (OCR) on the card.

Cards that store the card identification number (CID) and card specific data (CSD) in the payload memory are able to communicate this information only under data-transfer V_{DD} conditions. When the SDIO card host module and the card have incompatible V_{DD} ranges, the card is not able to complete the identification cycle and cannot send CSD data. For this purpose, the special commands, SEND_OP_COND (CMD1), SD_APP_OP_COND (ACMD41 for SD Memory), and IO_SEND_OP_COND (CMD5 for SD I/O), are designed to provide a mechanism to identify and reject cards that do not match the V_{DD} range desired by the SDIO card host. The SDIO card host sends the required V_{DD} voltage window as the operand of these commands. Cards that cannot perform data transfer in the specified range disconnect from the bus and go to the inactive state.

By using these commands without including the voltage range as the operand, the SDIO card host can query each card and determine the common voltage range before placing out-of-range cards in the inactive state. This query is used when the SDIO card host is able to select a common voltage range or when the user requires notification that cards are not usable.

26.4.4 Card identification process

The card identification process differs for MultiMediaCards and SD cards. For MultiMediaCard cards, the identification process starts at clock rate F_{od} . The SDIO_CMD line output drivers are open-drain and allow parallel card operation during this process. The registration process is accomplished as follows:

1. The bus is activated.
2. The SDIO card host broadcasts SEND_OP_COND (CMD1) to receive operation conditions.
3. The response is the wired AND operation of the operation condition registers from all cards.
4. Incompatible cards are placed in the inactive state.
5. The SDIO card host broadcasts ALL_SEND_CID (CMD2) to all active cards.
6. The active cards simultaneously send their CID numbers serially. Cards with outgoing CID bits that do not match the bits on the command line stop transmitting and must wait for the next identification cycle. One card successfully transmits a full CID to the SDIO card host and enters the Identification state.
7. The SDIO card host issues SET_RELATIVE_ADDR (CMD3) to that card. This new address is called the relative card address (RCA); it is shorter than the CID and addresses the card. The assigned card changes to the Standby state, it does not react to further identification cycles, and its output switches from open-drain to push-pull.
8. The SDIO card host repeats steps 5 through 7 until it receives a timeout condition.

For the SD card, the identification process starts at clock rate F_{od} , and the SDIO_CMD line output drives are push-pull drivers instead of open-drain. The registration process is accomplished as follows:

1. The bus is activated.
2. The SDIO card host broadcasts SD_APP_OP_COND (ACMD41).
3. The cards respond with the contents of their operation condition registers.
4. The incompatible cards are placed in the inactive state.
5. The SDIO card host broadcasts ALL_SEND_CID (CMD2) to all active cards.
6. The cards send back their unique card identification numbers (CIDs) and enter the Identification state.
7. The SDIO card host issues SET_RELATIVE_ADDR (CMD3) to an active card with an address. This new address is called the relative card address (RCA); it is shorter than the CID and addresses the card. The assigned card changes to the Standby state. The SDIO card host can reissue this command to change the RCA. The RCA of the card is the last assigned value.
8. The SDIO card host repeats steps 5 through 7 with all active cards.

For the SD I/O card, the registration process is accomplished as follows:

1. The bus is activated.
2. The SDIO card host sends `IO_SEND_OP_COND` (CMD5).
3. The cards respond with the contents of their operation condition registers.
4. The incompatible cards are set to the inactive state.
5. The SDIO card host issues `SET_RELATIVE_ADDR` (CMD3) to an active card with an address. This new address is called the relative card address (RCA); it is shorter than the CID and addresses the card. The assigned card changes to the Standby state. The SDIO card host can reissue this command to change the RCA. The RCA of the card is the last assigned value.

26.4.5 Block write

During block write (CMD24 - 27) one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. A card supporting block write is always able to accept a block of data defined by `WRITE_BL_LEN`. If the CRC fails, the card indicates the failure on the SDIO_D line and the transferred data are discarded and not written, and all further transmitted blocks (in multiple block write mode) are ignored.

If the host uses partial blocks whose accumulated length is not block aligned and, block misalignment is not allowed (CSD parameter `WRITE_BLK_MISALIGN` is not set), the card will detect the block misalignment error before the beginning of the first misaligned block. (`ADDRESS_ERROR` error bit is set in the status register). The write operation will also be aborted if the host tries to write over a write-protected area. In this case, however, the card will set the `WP_VIOLATION` bit.

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card reports an error and does not change any register contents. Some cards may require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card begins writing and holds the SDIO_D line low if its write buffer is full and unable to accept new data from a new `WRITE_BLOCK` command. The host may poll the status of the card with a `SEND_STATUS` command (CMD13) at any time, and the card will respond with its status. The `READY_FOR_DATA` status bit indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing CMD7 (to select a different card), which will place the card in the Disconnect state and release the SDIO_D line(s) without interrupting the write operation. When selecting the card again, it will reactivate busy indication by pulling SDIO_D to low if programming is still in progress and the write buffer is unavailable.

26.4.6 Block read

In Block read mode the basic unit of data transfer is a block whose maximum size is defined in the CSD (`READ_BL_LEN`). If `READ_BL_PARTIAL` is set, smaller blocks whose start and end addresses are entirely contained within one physical block (as defined by `READ_BL_LEN`) may also be transmitted. A CRC is appended to the end of each block, ensuring data transfer integrity. CMD17 (`READ_SINGLE_BLOCK`) initiates a block read and after completing the transfer, the card returns to the Transfer state.

CMD18 (`READ_MULTIPLE_BLOCK`) starts a transfer of several consecutive blocks.

The host can abort reading at any time, within a multiple block operation, regardless of its type. Transaction abort is done by sending the stop transmission command.

If the card detects an error (for example, out of range, address misalignment or internal error) during a multiple block read operation (both types) it stops the data transmission and remains in the data state. The host must then abort the operation by sending the stop transmission command. The read error is reported in the response to the stop transmission command.

If the host sends a stop transmission command after the card transmits the last block of a multiple block operation with a predefined number of blocks, it is responded to as an illegal command, since the card is no longer in the data state. If the host uses partial blocks whose accumulated length is not block-aligned and block misalignment is not allowed, the card detects a block misalignment error condition at the beginning of the first misaligned block (ADDRESS_ERROR error bit is set in the status register).

26.4.7 Stream access, stream write and stream read (MultiMediaCard only)

In stream mode, data is transferred in bytes and no CRC is appended at the end of each block.

Stream write (MultiMediaCard only)

WRITE_DAT_UNTIL_STOP (CMD20) starts the data transfer from the SDIO card host to the card, beginning at the specified address and continuing until the SDIO card host issues a stop command. When partial blocks are allowed (CSD parameter WRITE_BL_PARTIAL is set), the data stream can start and stop at any address within the card address space, otherwise it can only start and stop at block boundaries. Because the amount of data to be transferred is not determined in advance, a CRC cannot be used. When the end of the memory range is reached while sending data and no stop command is sent by the SD card host, any additional transferred data are discarded.

The maximum clock frequency for a stream write operation is given by the following equation fields of the card-specific data register:

$$\text{Maximumspeed} = \text{MIN}(\text{TRANSPEED}, \frac{(8 \times 2^{\text{writeblen}})(-\text{NSAC})}{\text{TAAC} \times \text{R2WFACTOR}})$$

- Maximumspeed = maximum write frequency
- TRANSPEED = maximum data transfer rate
- writeblen = maximum write data block length
- NSAC = data read access time 2 in CLK cycles
- TAAC = data read access time 1
- R2WFACTOR = write speed factor

If the host attempts to use a higher frequency, the card may not be able to process the data and stop programming, set the OVERRUN error bit in the status register, and while ignoring all further data transfer, wait (in the receive data state) for a stop command. The write operation is also aborted if the host tries to write over a write-protected area. In this case, however, the card sets the WP_VIOLATION bit.

Stream read (MultiMediaCard only)

`READ_DAT_UNTIL_STOP` (CMD11) controls a stream-oriented data transfer.

This command instructs the card to send its data, starting at a specified address, until the SDIO card host sends `STOP_TRANSMISSION` (CMD12). The stop command has an execution delay due to the serial command transmission and the data transfer stops after the end bit of the stop command. When the end of the memory range is reached while sending data and no stop command is sent by the SDIO card host, any subsequent data sent are considered undefined.

The maximum clock frequency for a stream read operation is given by the following equation and uses fields of the card specific data register.

$$\text{Maximumspeed} = \text{MIN}(\text{TRANSPEED}, \frac{(8 \times 2^{\text{readblen}}) (-\text{NSAC})}{\text{TAAC} \times \text{R2WFACTOR}})$$

- Maximumspeed = maximum read frequency
- TRANSPEED = maximum data transfer rate
- readblen = maximum read data block length
- writeblen = maximum write data block length
- NSAC = data read access time 2 in CLK cycles
- TAAC = data read access time 1
- R2WFACTOR = write speed factor

If the host attempts to use a higher frequency, the card is not able to sustain data transfer. If this happens, the card sets the `UNDERRUN` error bit in the status register, aborts the transmission and waits in the data state for a stop command.

26.4.8 Erase: group erase and sector erase

The erasable unit of the MultiMediaCard is the erase group. The erase group is measured in write blocks, which are the basic writable units of the card. The size of the erase group is a card-specific parameter and defined in the CSD.

The host can erase a contiguous range of Erase Groups. Starting the erase process is a three-step sequence.

First the host defines the start address of the range using the `ERASE_GROUP_START` (CMD35) command, next it defines the last address of the range using the `ERASE_GROUP_END` (CMD36) command and, finally, it starts the erase process by issuing the `ERASE` (CMD38) command. The address field in the erase commands is an Erase Group address in byte units. The card ignores all LSBs below the Erase Group size, effectively rounding the address down to the Erase Group boundary.

If an erase command is received out of sequence, the card sets the `ERASE_SEQ_ERROR` bit in the status register and resets the whole sequence.

If an out-of-sequence (neither of the erase commands, except `SEND_STATUS`) command received, the card sets the `ERASE_RESET` status bit in the status register, resets the erase sequence and executes the last command.

If the erase range includes write protected blocks, they are left intact and only unprotected blocks are erased. The `WP_ERASE_SKIP` status bit in the status register is set.

The card indicates that an erase is in progress by holding `SDIO_D` low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the card.

26.4.9 Wide bus selection or deselection

Wide bus (4-bit bus width) operation mode is selected or deselected using SET_BUS_WIDTH (ACMD6). The default bus width after power-up or GO_IDLE_STATE (CMD0) is 1 bit. SET_BUS_WIDTH (ACMD6) is only valid in a transfer state, which means that the bus width can be changed only after a card is selected by SELECT/DESELECT_CARD (CMD7).

26.4.10 Protection management

Three write protection methods for the cards are supported in the SDIO card host module:

1. internal card write protection (card responsibility)
2. mechanical write protection switch (SDIO card host module responsibility only)
3. password-protected card lock operation

Internal card write protection

Card data can be protected against write and erase. By setting the permanent or temporary write-protect bits in the CSD, the entire card can be permanently write-protected by the manufacturer or content provider. For cards that support write protection of groups of sectors by setting the WP_GRP_ENABLE bit in the CSD, portions of the data can be protected, and the write protection can be changed by the application. The write protection is in units of WP_GRP_SIZE sectors as specified in the CSD. The SET_WRITE_PROT and CLR_WRITE_PROT commands control the protection of the addressed group. The SEND_WRITE_PROT command is similar to a single block read command. The card sends a data block containing 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits. The address field in the write protect commands is a group address in byte units.

The card ignores all LSBs below the group size.

Mechanical write protect switch

A mechanical sliding tab on the side of the card allows the user to set or clear the write protection on a card. When the sliding tab is positioned with the window open, the card is write-protected, and when the window is closed, the card contents can be changed. A matched switch on the socket side indicates to the SDIO card host module that the card is write-protected. The SDIO card host module is responsible for protecting the card. The position of the write protect switch is unknown to the internal circuitry of the card.

Password protect

The password protection feature enables the SDIO card host module to lock and unlock a card with a password. The password is stored in the 128-bit PWD register and its size is set in the 8-bit PWD_LEN register. These registers are nonvolatile so that a power cycle does not erase them. Locked cards respond to and execute certain commands. This means that the SDIO card host module is allowed to reset, initialize, select, and query for status, however it is not allowed to access data on the card. When the password is set (as indicated by a nonzero value of PWD_LEN), the card is locked automatically after power-up. As with the CSD and CID register write commands, the lock/unlock commands are available in the transfer state only. In this state, the command does not include an address argument and the card must be selected before using it. The card lock/unlock commands have the structure and bus transaction types of a regular single-block write command. The transferred data block includes all of the required information for the command (the

password setting mode, the PWD itself, and card lock/unlock). The command data block size is defined by the SDIO card host module before it sends the card lock/unlock command, and has the structure shown in [Table 124](#).

The bit settings are as follows:

- ERASE: setting it forces an erase operation. All other bits must be zero, and only the command byte is sent
- LOCK_UNLOCK: setting it locks the card. LOCK_UNLOCK can be set simultaneously with SET_PWD, however not with CLR_PWD
- CLR_PWD: setting it clears the password data
- SET_PWD: setting it saves the password data to memory
- PWD_LEN: it defines the length of the password in bytes
- PWD: the password (new or currently used, depending on the command)

The following sections list the command sequences to set/reset a password, lock/unlock the card, and force an erase.

Setting the password

1. Select a card (SELECT/DESELECT_CARD, CMD7), if none is already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit card lock/unlock mode, the 8-bit PWD_LEN, and the number of bytes of the new password. When a password replacement is done, the block size must take into account that both the old and the new passwords are sent with the command.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (SET_PWD = 1), the length (PWD_LEN), and the password (PWD) itself. When a password replacement is done, the length value (PWD_LEN) includes the length of both passwords, the old and the new one, and the PWD field includes the old password (currently used) followed by the new password.
4. When the password is matched, the new password and its size are saved into the PWD and PWD_LEN fields, respectively. When the old password sent does not correspond (in size and/or content) to the expected password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the password is not changed.

The password length field (PWD_LEN) indicates whether a password is currently set. When this field is nonzero, there is a password set and the card locks itself after power-up. It is possible to lock the card immediately in the current power session by setting the LOCK_UNLOCK bit (while setting the password) or sending an additional command for card locking.

Resetting the password

1. Select a card (SELECT/DESELECT_CARD, CMD7), if none is already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit card lock/unlock mode, the 8-bit PWD_LEN, and the number of bytes in the currently used password.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (CLR_PWD = 1), the length (PWD_LEN) and the password (PWD) itself. The LOCK_UNLOCK bit is ignored.
4. When the password is matched, the PWD field is cleared and PWD_LEN is set to 0. When the password sent does not correspond (in size and/or content) to the expected

password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the password is not changed.

Locking a card

1. Select a card (SELECT/DESELECT_CARD, CMD7), if none is already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit card lock/unlock mode (byte 0 in [Table 124](#)), the 8-bit PWD_LEN, and the number of bytes of the current password.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (LOCK_UNLOCK = 1), the length (PWD_LEN), and the password (PWD) itself.
4. When the password is matched, the card is locked and the CARD_IS_LOCKED status bit is set in the card status register. When the password sent does not correspond (in size and/or content) to the expected password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the lock fails.

It is possible to set the password and to lock the card in the same sequence. In this case, the SDIO card host module performs all the required steps for setting the password (see [Setting the password on page 731](#)), however it is necessary to set the LOCK_UNLOCK bit in Step 3 when the new password command is sent.

When the password is previously set (PWD_LEN is not 0), the card is locked automatically after power on reset. An attempt to lock a locked card or to lock a card that does not have a password fails and the LOCK_UNLOCK_FAILED error bit is set in the card status register.

Unlocking the card

1. Select a card (SELECT/DESELECT_CARD, CMD7), if none is already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit cardlock/unlock mode (byte 0 in [Table 124](#)), the 8-bit PWD_LEN, and the number of bytes of the current password.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (LOCK_UNLOCK = 0), the length (PWD_LEN), and the password (PWD) itself.
4. When the password is matched, the card is unlocked and the CARD_IS_LOCKED status bit is cleared in the card status register. When the password sent is not correct in size and/or content and does not correspond to the expected password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the card remains locked.

The unlocking function is only valid for the current power session. When the PWD field is not clear, the card is locked automatically on the next power-up.

An attempt to unlock an unlocked card fails and the LOCK_UNLOCK_FAILED error bit is set in the card status register.

Forcing erase

If the user has forgotten the password (PWD content), it is possible to access the card after clearing all the data on the card. This forced erase operation erases all card data and all password data.

1. Select a card (SELECT/DESELECT_CARD, CMD7), if none is already selected.
2. Set the block length (SET_BLOCKLEN, CMD16) to 1 byte. Only the 8-bit card lock/unlock byte (byte 0 in [Table 124](#)) is sent.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data byte on the data line including the 16-bit CRC. The data block indicates the mode (ERASE = 1). All other bits must be zero.
4. When the ERASE bit is the only bit set in the data field, all card contents are erased, including the PWD and PWD_LEN fields, and the card is no longer locked. When any other bits are set, the LOCK_UNLOCK_FAILED error bit is set in the card status register and the card retains all of its data, and remains locked.

An attempt to use a force erase on an unlocked card fails and the LOCK_UNLOCK_FAILED error bit is set in the card status register.

26.4.11 Card status register

The response format R1 contains a 32-bit field named card status. This field is intended to transmit the card status information (which may be stored in a local status register) to the host. If not specified otherwise, the status entries are always related to the previously issued command.

[Table 111](#) defines the different entries of the status. The type and clear condition fields in the table are abbreviated as follows:

Type:

- E: error bit
- S: status bit
- R: detected and set for the actual command response
- X: detected and set during command execution. The SDIO card host must poll the card by issuing the status command to read these bits.

Clear condition:

- A: according to the card current state
- B: always related to the previous command. Reception of a valid command clears it (with a delay of one command)
- C: clear by read

Table 111. Card status

Bits	Identifier	Type	Value	Description	Clear condition
31	ADDRESS_OUT_OF_RANGE	E R X	'0'= no error '1'= error	The command address argument was out of the allowed range for this card. A multiple block or stream read/write operation is (although started in a valid address) attempting to read or write beyond the card capacity.	C
30	ADDRESS_MISALIGN		'0'= no error '1'= error	The commands address argument (in accordance with the currently set block length) positions the first data block misaligned to the card physical blocks. A multiple block read/write operation (although started with a valid address/block-length combination) is attempting to read or write a data block which is not aligned with the physical blocks of the card.	C
29	BLOCK_LEN_ERROR		'0'= no error '1'= error	Either the argument of a SET_BLOCKLEN command exceeds the maximum value allowed for the card, or the previously defined block length is illegal for the current command (e.g. the host issues a write command, the current block length is smaller than the maximum allowed value for the card and it is not allowed to write partial blocks)	C
28	ERASE_SEQ_ERROR		'0'= no error '1'= error	An error in the sequence of erase commands occurred.	C
27	ERASE_PARAM	E X	'0'= no error '1'= error	An invalid selection of erase groups for erase occurred.	C
26	WP_VIOLATION	E X	'0'= no error '1'= error	Attempt to program a write-protected block.	C
25	CARD_IS_LOCKED	S R	'0' = card unlocked '1' = card locked	When set, signals that the card is locked by the host	A
24	LOCK_UNLOCK_FAILED	E X	'0'= no error '1'= error	Set when a sequence or password error has been detected in lock/unlock card command	C
23	COM_CRC_ERROR	E R	'0'= no error '1'= error	The CRC check of the previous command failed.	B
22	ILLEGAL_COMMAND	E R	'0'= no error '1'= error	Command not legal for the card state	B
21	CARD_ECC_FAILED	E X	'0'= success '1'= failure	Card internal ECC was applied but failed to correct the data.	C
20	CC_ERROR	E R	'0'= no error '1'= error	(Undefined by the standard) A card error occurred, which is not related to the host command.	C

Table 111. Card status (continued)

Bits	Identifier	Type	Value	Description	Clear condition
19	ERROR	E X	'0'= no error '1'= error	(Undefined by the standard) A generic card error related to the (and detected during) execution of the last host command (e.g. read or write failures).	C
18	Reserved				
17	Reserved				
16	CID/CSD_OVERWRITE	E X	'0'= no error '1'= error	Can be either of the following errors: – The CID register has already been written and cannot be overwritten – The read-only section of the CSD does not match the card contents – An attempt to reverse the copy (set as original) or permanent WP (unprotected) bits was made	C
15	WP_ERASE_SKIP	E X	'0'= not protected '1'= protected	Set when only partial address space was erased due to existing write	C
14	CARD_ECC_DISABLED	S X	'0'= enabled '1'= disabled	The command has been executed without using the internal ECC.	A
13	ERASE_RESET		'0'= cleared '1'= set	An erase sequence was cleared before executing because an out of erase sequence command was received (commands other than CMD35, CMD36, CMD38 or CMD13)	C
12:9	CURRENT_STATE	S R	0 = Idle 1 = Ready 2 = Ident 3 = Stby 4 = Tran 5 = Data 6 = Rcv 7 = Prg 8 = Dis 9 = Bst 10-15 = reserved	The state of the card when receiving the command. If the command execution causes a state change, it will be visible to the host in the response on the next command. The four bits are interpreted as a binary number between 0 and 15.	B
8	READY_FOR_DATA	S R	'0'= not ready '1'= ready	Corresponds to buffer empty signalling on the bus	
7	SWITCH_ERROR	E X	'0'= no error '1'= switch error	If set, the card did not switch to the expected mode as requested by the SWITCH command	B
6	Reserved				
5	APP_CMD	S R	'0' = Disabled '1' = Enabled	The card will expect ACMD, or an indication that the command has been interpreted as ACMD	C
4	Reserved for SD I/O Card				

Table 111. Card status (continued)

Bits	Identifier	Type	Value	Description	Clear condition
3	AKE_SEQ_ERROR	E R	'0'= no error '1'= error	Error in the sequence of the authentication process	C
2	Reserved for application specific commands				
1	Reserved for manufacturer test mode				
0					

26.4.12 SD status register

The SD status contains status bits that are related to the SD memory card proprietary features and may be used for future application-specific usage. The size of the SD Status is one data block of 512 bits. The contents of this register are transmitted to the SDIO card host if ACMD13 is sent (CMD55 followed with CMD13). ACMD13 can be sent to a card in transfer state only (card is selected).

Table 112 defines the different entries of the SD status register. The type and clear condition fields in the table are abbreviated as follows:

Type:

- E: error bit
- S: status bit
- R: detected and set for the actual command response
- X: detected and set during command execution. The SDIO card Host must poll the card by issuing the status command to read these bits

Clear condition:

- A: according to the card current state
- B: always related to the previous command. Reception of a valid command clears it (with a delay of one command)
- C: clear by read

Table 112. SD status

Bits	Identifier	Type	Value	Description	Clear condition
511: 510	DAT_BUS_WIDTH	S R	'00'= 1 (default) '01'= reserved '10'= 4 bit width '11'= reserved	Shows the currently defined databus width that was defined by SET_BUS_WIDTH command	A
509	SECURED_MODE	S R	'0'= Not in the mode '1'= In Secured Mode	Card is in Secured Mode of operation (refer to the "SD Security Specification").	A
508: 496	Reserved				

Table 112. SD status (continued)

Bits	Identifier	Type	Value	Description	Clear condition
495: 480	SD_CARD_TYPE	S R	'00xxh'= SD Memory Cards as defined in Physical Spec Ver1.01-2.00 ('x'= don't care). The following cards are currently defined: '0000'= Regular SD RD/WR Card. '0001'= SD ROM Card	In the future, the 8 LSBs will be used to define different variations of an SD memory card (each bit will define different SD types). The 8 MSBs will be used to define SD Cards that do not comply with current SD physical layer specification.	A
479: 448	SIZE_OF_PROTECTED_AREA	S R	Size of protected area (See below)	(See below)	A
447: 440	SPEED_CLASS	S R	Speed Class of the card (See below)	(See below)	A
439: 432	PERFORMANCE_MOVE	S R	Performance of move indicated by 1 [MB/s] step. (See below)	(See below)	A
431:428	AU_SIZE	S R	Size of AU (See below)	(See below)	A
427:424	Reserved				
423:408	ERASE_SIZE	S R	Number of AUs to be erased at a time	(See below)	A
407:402	ERASE_TIMEOUT	S R	Timeout value for erasing areas specified by UNIT_OF_ERASE_AU	(See below)	A
401:400	ERASE_OFFSET	S R	Fixed offset value added to erase time.	(See below)	A
399:312	Reserved				
311:0	Reserved for Manufacturer				

SIZE_OF_PROTECTED_AREA

Setting this field differs between standard- and high-capacity cards. In the case of a standard-capacity card, the capacity of protected area is calculated as follows:

$$\text{Protected area} = \text{SIZE_OF_PROTECTED_AREA_} * \text{MULT} * \text{BLOCK_LEN}.$$

SIZE_OF_PROTECTED_AREA is specified by the unit in MULT*BLOCK_LEN.

In the case of a high-capacity card, the capacity of protected area is specified in this field:

$$\text{Protected area} = \text{SIZE_OF_PROTECTED_AREA}$$

SIZE_OF_PROTECTED_AREA is specified by the unit in bytes.

SPEED_CLASS

This 8-bit field indicates the speed class and the value can be calculated by $P_W/2$ (where P_W is the write performance).

Table 113. Speed class code field

SPEED_CLASS	Value definition
00h	Class 0
01h	Class 2
02h	Class 4
03h	Class 6
04h – FFh	Reserved

PERFORMANCE_MOVE

This 8-bit field indicates Pm (performance move) and the value can be set by 1 [MB/sec] steps. If the card does not move used RUs (recording units), Pm should be considered as infinity. Setting the field to FFh means infinity.

Table 114. Performance move field

PERFORMANCE_MOVE	Value definition
00h	Not defined
01h	1 [MB/sec]
02h	02h 2 [MB/sec]
-----	-----
FEh	254 [MB/sec]
FFh	Infinity

AU_SIZE

This 4-bit field indicates the AU size and the value can be selected in the power of 2 base from 16 KB.

Table 115. AU_SIZE field

AU_SIZE	Value definition
00h	Not defined
01h	16 KB
02h	32 KB
03h	64 KB
04h	128 KB
05h	256 KB
06h	512 KB
07h	1 MB
08h	2 MB

Table 115. AU_SIZE field (continued)

AU_SIZE	Value definition
09h	4 MB
Ah – Fh	Reserved

The maximum AU size, which depends on the card capacity, is defined in [Table 116](#). The card can be set to any AU size between RU size and maximum AU size.

Table 116. Maximum AU size

Capacity	16 MB-64 MB	128 MB-256 MB	512 MB	1 GB-32 GB
Maximum AU Size	512 KB	1 MB	2 MB	4 MB

ERASE_SIZE

This 16-bit field indicates NERASE. When NERASE numbers of AUs are erased, the timeout value is specified by ERASE_TIMEOUT (Refer to [ERASE_TIMEOUT](#)). The host should determine the proper number of AUs to be erased in one operation so that the host can show the progress of the erase operation. If this field is set to 0, the erase timeout calculation is not supported.

Table 117. Erase size field

ERASE_SIZE	Value definition
0000h	Erase timeout calculation is not supported.
0001h	1 AU
0002h	2 AU
0003h	3 AU
-----	-----
FFFFh	65535 AU

ERASE_TIMEOUT

This 6-bit field indicates TERASE and the value indicates the erase timeout from offset when multiple AUs are being erased as specified by ERASE_SIZE. The range of ERASE_TIMEOUT can be defined as up to 63 seconds and the card manufacturer can choose any combination of ERASE_SIZE and ERASE_TIMEOUT depending on the implementation. Determining ERASE_TIMEOUT determines the ERASE_SIZE.

Table 118. Erase timeout field

ERASE_TIMEOUT	Value definition
00	Erase timeout calculation is not supported.
01	1 [sec]
02	2 [sec]
03	3 [sec]

Table 118. Erase timeout field (continued)

ERASE_TIMEOUT	Value definition
-----	-----
63	63 [sec]

ERASE_OFFSET

This 2-bit field indicates T_{OFFSET} and one of four values can be selected. This field is meaningless if the ERASE_SIZE and ERASE_TIMEOUT fields are set to 0.

Table 119. Erase offset field

ERASE_OFFSET	Value definition
0h	0 [sec]
1h	1 [sec]
2h	2 [sec]
3h	3 [sec]

26.4.13 SD I/O mode**SD I/O interrupts**

To allow the SD I/O card to interrupt the MultiMediaCard/SD module, an interrupt function is available on a pin on the SD interface. Pin 8, used as SDIO_D1 when operating in the 4-bit SD mode, signals the cards interrupt to the MultiMediaCard/SD module. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the MultiMediaCard/SD module or deasserted due to the end of the interrupt period. After the MultiMediaCard/SD module has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card's internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDIO_D[3:0]). The MultiMediaCard/SD module samples the level of pin 8 (SDIO_D/IRQ) into the interrupt detector only during the interrupt period. At all other times, the MultiMediaCard/SD module ignores this value.

The interrupt period is applicable for both memory and I/O operations. The definition of the interrupt period for operations with single blocks is different from the definition for multiple-block data transfers.

SD I/O suspend and resume

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the MMC/SD bus. To share access to the MMC/SD module among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the MMC/SD module can temporarily halt a data transfer operation to one function or memory (suspend) to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is resumed (restarted) where it left off. Support of suspend/resume is optional on a per-card basis. To perform the

suspend/resume operation on the MMC/SD bus, the MMC/SD module performs the following steps:

1. Determines the function currently using the SDIO_D [3:0] line(s)
2. Requests the lower-priority or slower transaction to suspend
3. Waits for the transaction suspension to complete
4. Begins the higher-priority transaction
5. Waits for the completion of the higher priority transaction
6. Restores the suspended transaction

SD I/O ReadWait

The optional ReadWait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The ReadWait operation allows the MMC/SD module to signal a card that it is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the MMC/SD module to send commands to any function within the SD I/O device. To determine when a card supports the ReadWait protocol, the MMC/SD module must test capability bits in the internal card registers. The timing for ReadWait is based on the interrupt period.

26.4.14 Commands and responses

Application-specific and general commands

The SD card host module system is designed to provide a standard interface for a variety of applications types. In this environment, there is a need for specific customer/application features. To implement these features, two types of generic commands are defined in the standard: application-specific commands (ACMD) and general commands (GEN_CMD).

When the card receives the APP_CMD (CMD55) command, the card expects the next command to be an application-specific command. ACMDs have the same structure as regular MultiMediaCard commands and can have the same CMD number. The card recognizes it as ACMD because it appears after APP_CMD (CMD55). When the command immediately following the APP_CMD (CMD55) is not a defined application-specific command, the standard command is used. For example, when the card has a definition for SD_STATUS (ACMD13), and receives CMD13 immediately following APP_CMD (CMD55), this is interpreted as SD_STATUS (ACMD13). However, when the card receives CMD7 immediately following APP_CMD (CMD55) and the card does not have a definition for ACMD7, this is interpreted as the standard (SELECT/DESELECT_CARD) CMD7.

To use one of the manufacturer-specific ACMDs the SD card Host must perform the following steps:

1. Send APP_CMD (CMD55)
The card responds to the MultiMediaCard/SD module, indicating that the APP_CMD bit is set and an ACMD is now expected.
2. Send the required ACMD
The card responds to the MultiMediaCard/SD module, indicating that the APP_CMD bit is set and that the accepted command is interpreted as an ACMD. When a nonACMD is sent, it is handled by the card as a normal MultiMediaCard command and the APP_CMD bit in the card status register stays clear.

When an invalid command is sent (neither ACMD nor CMD) it is handled as a standard MultiMediaCard illegal command error.

The bus transaction for a GEN_CMD is the same as the single-block read or write commands (WRITE_BLOCK, CMD24 or READ_SINGLE_BLOCK,CMD17). In this case, the argument denotes the direction of the data transfer rather than the address, and the data block has vendor-specific format and meaning.

The card must be selected (in transfer state) before sending GEN_CMD (CMD56). The data block size is defined by SET_BLOCKLEN (CMD16). The response to GEN_CMD (CMD56) is in R1b format.

Command types

Both application-specific and general commands are divided into the four following types:

- **broadcast command (BC)**: sent to all cards; no responses returned.
- **broadcast command with response (BCR)**: sent to all cards; responses received from all cards simultaneously.
- **addressed (point-to-point) command (AC)**: sent to the card that is selected; does not include a data transfer on the SDIO_D line(s).
- **addressed (point-to-point) data transfer command (ADTC)**: sent to the card that is selected; includes a data transfer on the SDIO_D line(s).

Command formats

See [Table 104 on page 718](#) for command formats.

Commands for the MultiMediaCard/SD module

Table 120. Block-oriented write commands

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD23	ac	[31:16] set to 0 [15:0] number of blocks	R1	SET_BLOCK_COUNT	Defines the number of blocks which are going to be transferred in the multiple-block read or write command that follows.
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows or the requested number of blocks has been received.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command must be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.

Table 121. Block-oriented write protection commands

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card-specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				

Table 122. Erase commands

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD32 ... CMD34		Reserved. These command indexes cannot be used in order to maintain backward compatibility with older versions of the MultiMediaCard.			
CMD35	ac	[31:0] data address	R1	ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37		Reserved. This command index cannot be used in order to maintain backward compatibility with older versions of the MultiMediaCards			
CMD38	ac	[31:0] stuff bits	R1	ERASE	Erases all previously selected write blocks.

Table 123. I/O mode commands

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD39	ac	[31:16] RCA [15:15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card and a register and provides the data for writing if the write flag is set. The R4 response contains data read from the addressed register. This command accesses application-dependent registers that are not defined in the MultiMediaCard standard.

Table 123. I/O mode commands (continued)

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Places the system in the interrupt mode.
CMD41	Reserved				

Table 124. Lock card

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Sets/resets the password or locks/unlocks the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43 ... CMD54	Reserved				

Table 125. Application-specific commands

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command bits is an application specific command rather than a standard command
CMD56	adtc	[31:1] stuff bits [0]: RD/WR			Used either to transfer a data block to the card or to get a data block from the card for general purpose/application-specific commands. The size of the data block shall be set by the SET_BLOCK_LEN command.
CMD57 ... CMD59	Reserved.				
CMD60 ... CMD63	Reserved for manufacturer.				

26.5 Response formats

All responses are sent via the MCCMD command line SDIO_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. All responses, except for the R3 response type, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

There are five types of responses. Their formats are defined as follows:

26.5.1 R1 (normal response command)

Code length = 48 bits. The 45:40 bits indicate the index of the command to be responded to, this value being interpreted as a binary-coded number (between 0 and 63). The status of the card is coded in 32 bits.

Table 126. R1 response

Bit position	Width (bits)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	X	Command index
[39:8]	32	X	Card status
[7:1]	7	X	CRC7
0	1	1	End bit

26.5.2 R1b

It is identical to R1 with an optional busy signal transmitted on the data line. The card may become busy after receiving these commands based on its state prior to the command reception.

26.5.3 R2 (CID, CSD register)

Code length = 136 bits. The contents of the CID register are sent as a response to the CMD2 and CMD10 commands. The contents of the CSD register are sent as a response to CMD9. Only the bits [127...1] of the CID and CSD are transferred, the reserved bit [0] of these registers is replaced by the end bit of the response. The card indicates that an erase is in progress by holding MCDAT low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the card.

Table 127. R2 response

Bit position	Width (bits)	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	'111111'	Command index
[127:1]	127	X	Card status
0	1	1	End bit

26.5.4 R3 (OCR register)

Code length: 48 bits. The contents of the OCR register are sent as a response to CMD1. The level coding is as follows: restricted voltage windows = low, card busy = low.

Table 128. R3 response

Bit position	Width (bits)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	'111111'	Reserved
[39:8]	32	X	OCR register
[7:1]	7	'1111111'	Reserved
0	1	1	End bit

26.5.5 R4 (Fast I/O)

Code length: 48 bits. The argument field contains the RCA of the addressed card, the register address to be read out or written to, and its content.

Table 129. R4 response

Bit position	Width (bits)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	'111111'	Reserved
[39:8] Argument field	[31:16]	16	RCA
	[15:8]	8	register address
	[7:0]	8	read register contents
	[7:1]	7	'1111111'
0	1	1	End bit

26.5.6 R4b

For SD I/O only: an SDIO card receiving the CMD5 will respond with a unique SDIO response R4. The format is:

Table 130. R4b response

Bit position	Width (bits)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Reserved

Table 130. R4b response (continued)

Bit position	Width (bits)	Value	Description
[39:8] Argument field	39	16	X Card is ready
	[38:36]	3	X Number of I/O functions
	35	1	X Present memory
	[34:32]	3	X Stuff bits
	[31:8]	24	X I/O ORC
[7:1]	7	X Reserved	
0	1	1	End bit

Once an SD I/O card has received a CMD5, the I/O portion of that card is enabled to respond normally to all further commands. This I/O enable of the function within the I/O card will remain set until a reset, power cycle or CMD52 with write to I/O reset is received by the card. Note that an SD memory-only card may respond to a CMD5. The proper response for a memory-only card would be *Present memory* = 1 and *Number of I/O functions* = 0. A memory-only card built to meet the SD Memory Card specification version 1.0 would detect the CMD5 as an illegal command and not respond. The I/O aware host will send CMD5. If the card responds with response R4, the host determines the card's configuration based on the data contained within the R4 response.

26.5.7 R5 (interrupt request)

Only for MultiMediaCard. Code length: 48 bits. If the response is generated by the host, the RCA field in the argument will be 0x0.

Table 131. R5 response

Bit position	Width (bits)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	'111111'	CMD40
[39:8] Argument field	[31:16]	16	X RCA [31:16] of winning card or of the host
	[15:0]	16	X Not defined. May be used for IRQ data
[7:1]	7	X CRC7	
0	1	1	End bit

26.5.8 R6

Only for SD I/O. The normal response to CMD3 by a memory device. It is shown in [Table 132](#).

Table 132. R6 response

Bit position	Width (bits)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	'101000'	CMD40
[39:8] Argument field	[31:16]	X	RCA [31:16] of winning card or of the host
	[15:0]	X	Not defined. May be used for IRQ data
[7:1]	7	X	CRC7
0	1	1	End bit

The card [23:8] status bits are changed when CMD3 is sent to an I/O-only card. In this case, the 16 bits of response are the SD I/O-only values:

- Bit [15] COM_CRC_ERROR
- Bit [14] ILLEGAL_COMMAND
- Bit [13] ERROR
- Bits [12:0] Reserved

26.6 SDIO I/O card-specific operations

The following features are SD I/O-specific operations:

- SDIO read wait operation by SDIO_D2 signalling
- SDIO read wait operation by stopping the clock
- SDIO suspend/resume operation (write and read suspend)
- SDIO interrupts

The SDIO supports these operations only if the SDIO_DCTRL[11] bit is set, except for read suspend that does not need specific hardware implementation.

26.6.1 SDIO I/O read wait operation by SDIO_D2 signalling

It is possible to start the readwait interval before the first block is received: when the data path is enabled (SDIO_DCTRL[0] bit set), the SDIO-specific operation is enabled (SDIO_DCTRL[11] bit set), read wait starts (SDIO_DCTRL[10]=0 and SDI_DCTRL[8]=1) and data direction is from card to SDIO (SDIO_DCTRL[1]=1), the DPSM directly moves from Idle to Readwait. In Readwait the DPSM drives SDIO_D2 to 0 after 2 SDIO_CK clock cycles. In this state, when you set the RWSTOP bit (SDIO_DCTRL[9]), the DPSM remains in Wait for two more SDIO_CK clock cycles to drive SDIO_D2 to 1 for one clock cycle (in accordance with SDIO specification). The DPSM then starts waiting again until it receives data from the card. The DPSM will not start a readwait interval while receiving a block even if read wait start is set: the readwait interval will start after the CRC is received. The RWSTOP bit has to be cleared to start a new read wait operation. During the readwait interval, the SDIO can detect SDIO interrupts on SDIO_D1.

26.6.2 SDIO read wait operation by stopping SDIO_CK

If the SDIO card does not support the previous read wait method, the SDIO can perform a read wait by stopping SDIO_CK (SDIO_DCTRL is set just like in the method presented in [Section 26.6.1](#), but SDIO_DCTRL[10] =1): DPSM stops the clock two SDIO_CK cycles after the end bit of the current received block and starts the clock again after the read wait start bit is set.

As SDIO_CK is stopped, any command can be issued to the card. During a read/wait interval, the SDIO can detect SDIO interrupts on SDIO_D1.

26.6.3 SDIO suspend/resume operation

While sending data to the card, the SDIO can suspend the write operation. the SDIO_CMD[11] bit is set and indicates to the CPSM that the current command is a suspend command. The CPSM analyzes the response and when the ACK is received from the card (suspend accepted), it acknowledges the DPSM that goes Idle after receiving the CRC token of the current block.

The hardware does not save the number of the remaining block to be sent to complete the suspended operation (resume).

The write operation can be suspended by software, just by disabling the DPSM (SDIO_DCTRL[0] =0) when the ACK of the suspend command is received from the card. The DPSM enters then the Idle state.

To suspend a read: the DPSM waits in the Wait_r state as the function to be suspended sends a complete packet just before stopping the data transaction. The application continues reading RxFIFO until the FIFO is empty, and the DPSM goes Idle automatically.

26.6.4 SDIO interrupts

SDIO interrupts are detected on the SDIO_D1 line once the SDIO_DCTRL[11] bit is set.

26.7 CE-ATA specific operations

The following features are CE-ATA specific operations:

- sending the command completion signal disable to the CE-ATA device
- receiving the command completion signal from the CE-ATA device
- signaling the completion of the CE-ATA command to the CPU, using the status bit and/or interrupt.

The SDIO supports these operations only for the CE-ATA CMD61 command, that is, if SDIO_CMD[14] is set.

26.7.1 Command completion signal disable

Command completion signal disable is sent 8 bit cycles after the reception of a **short** response if the ‘enable CMD completion’ bit, SDIO_CMD[12], is not set and the ‘not interrupt Enable’ bit, SDIO_CMD[13], is set.

The CPSM enters the Pend state, loading the command shift register with the disable sequence “00001” and, the command counter with 43. Eight cycles after, a trigger moves

the CPSM to the Send state. When the command counter reaches 48, the CPSM becomes Idle as no response is awaited.

26.7.2 Command completion signal enable

If the ‘enable CMD completion’ bit SDIO_CMD[12] is set and the ‘not interrupt Enable’ bit SDIO_CMD[13] is set, the CPSM waits for the command completion signal in the Waitcpl state.

When ‘0’ is received on the CMD line, the CPSM enters the Idle state. No new command can be sent for 7 bit cycles. Then, for the last 5 cycles (out of the 7) the CMD line is driven to ‘1’ in push-pull mode.

26.7.3 CE-ATA interrupt

The command completion is signaled to the CPU by the status bit SDIO_STA[23]. This static bit can be cleared with the clear bit SDIO_ICR[23].

The SDIO_STA[23] status bit can generate an interrupt on each interrupt line, depending on the mask bit SDIO_MASKx[23].

26.7.4 Aborting CMD61

If the command completion disable signal has not been sent and CMD61 needs to be aborted, the command state machine must be disabled. It then becomes Idle, and the CMD12 command can be sent. No command completion disable signal is sent during the operation.

26.8 HW flow control

The HW flow control functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDIO_CK and freeze SDIO state machines. The data transfer is stalled while the FIFO is unable to transmit or receive data. Only state machines clocked by SDIOCLK are frozen, the APB2 interface is still alive. The FIFO can thus be filled or emptied even if flow control is activated.

To enable HW flow control, the SDIO_CLKCR[14] register bit must be set to 1. After reset Flow Control is disabled.

26.9 SDIO registers

The device communicates to the system via 32-bit-wide control registers accessible via APB2.

26.9.1 SDIO power control register (SDIO_POWER)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value

Bits 1:0 **PWRCTRL:** Power supply control bits.

These bits are used to define the current functional state of the card clock:

00: Power-off: the clock to card is stopped.

01: Reserved

10: Reserved power-up

11: Power-on: the card is clocked.

Note: At least seven HCLK clock periods are needed between two write accesses to this register.

Note: After a data write, data cannot be written to this register for three SDIOCLK (48 MHz) clock periods plus two PCLK2 clock periods.

26.9.2 SDI clock control register (SDIO_CLKCR)

Address offset: 0x04

Reset value: 0x0000 0000

The SDIO_CLKCR register controls the SDIO_CK output clock.

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																															
rw																															

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

Bit 14 **HWFC_EN:** HW Flow Control enable

0b: HW Flow Control is disabled

1b: HW Flow Control is enabled

When HW Flow Control is enabled, the meaning of the TXFIFOE and RXFIFOF interrupt signals, please see SDIO Status register definition in [Section 26.9.11](#).

Bit 13 **NEGEDGE:** SDIO_CK dephasing selection bit

0b: SDIO_CK generated on the rising edge of the master clock SDIOCLK

1b: SDIO_CK generated on the falling edge of the master clock SDIOCLK

Bits 12:11 **WIDBUS:** Wide bus mode enable bit
 00: Default bus mode: SDIO_D0 used
 01: 4-wide bus mode: SDIO_D[3:0] used
 10: 8-wide bus mode: SDIO_D[7:0] used

Bit 10 **BYPASS:** Clock divider bypass enable bit
 0: Disable bypass: SDIOCLK is divided according to the CLKDIV value before driving the SDIO_CK output signal.
 1: Enable bypass: SDIOCLK directly drives the SDIO_CK output signal.

Bit 9 **PWRSAV:** Power saving configuration bit
 For power saving, the SDIO_CK clock output can be disabled when the bus is idle by setting PWRSAV:
 0: SDIO_CK clock is always enabled
 1: SDIO_CK is only enabled when the bus is active

Bit 8 **CLKEN:** Clock enable bit
 0: SDIO_CK is disabled
 1: SDIO_CK is enabled

Bits 7:0 **CLKDIV:** Clock divide factor
 This field defines the divide factor between the input clock (SDIOCLK) and the output clock (SDIO_CK): SDIO_CK frequency = SDIOCLK / [CLKDIV + 2].

- Note:**
- 1 While the SD/SDIO card or MultiMediaCard is in identification mode, the SDIO_CK frequency must be less than 400 kHz.
 - 2 The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.
 - 3 After a data write, data cannot be written to this register for three SDIOCLK (48 MHz) clock periods plus two PCLK2 clock periods. SDIO_CK can also be stopped during the read wait interval for SD I/O cards: in this case the SDIO_CLKCR register does not control SDIO_CK.

26.9.3 SDIO argument register (SDIO_ARG)

Address offset: 0x08

Reset value: 0x0000 0000

The SDIO_ARG register contains a 32-bit command argument, which is sent to a card as part of a command message.

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
CMDARG																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:0 **CMDARG:** Command argument

Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

26.9.4 SDIO command register (SDIO_CMD)

Address offset: 0x0C

Reset value: 0x0000 0000

The SDIO_CMD register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

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														CE-ATACMD	nIEN	ENCMDcompl	SDIOSuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP	CMDINDEX									
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 31:15 Reserved, must be kept at reset value

Bit 14 **ATACMD:** CE-ATA command

If ATACMD is set, the CPSM transfers CMD61.

Bit 13 **nIEN:** not Interrupt Enable

If this bit is 0, interrupts in the CE-ATA device are enabled.

Bit 12 **ENCMDcompl:** Enable CMD completion

If this bit is set, the command completion signal is enabled.

Bit 11 **SDIOSuspend:** SD I/O suspend command

If this bit is set, the command to be sent is a suspend command (to be used only with SDIO card).

Bit 10 **CPSMEN:** Command path state machine (CPSM) Enable bit

If this bit is set, the CPSM is enabled.

Bit 9 **WAITPEND:** CPSM Waits for ends of data transfer (CmdPend internal signal).

If this bit is set, the CPSM waits for the end of data transfer before it starts sending a command.

Bit 8 **WAITINT:** CPSM waits for interrupt request

If this bit is set, the CPSM disables command timeout and waits for an interrupt request.

Bits 7:6 **WAITRESP:** Wait for response bits

They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.

00: No response, expect CMDSENT flag

01: Short response, expect CMDREND or CCRCFAIL flag

10: No response, expect CMDSENT flag

11: Long response, expect CMDREND or CCRCFAIL flag

Bit 5:0 **CMDINDEX:** Command index

The command index is sent to the card as part of a command message.

Note: 1 After a data write, data cannot be written to this register for three SDIOCLK (48 MHz) clock periods plus two PCLK2 clock periods.

2 MultiMediaCards can send two kinds of response: short responses, 48 bits long, or long responses, 136 bits long. SD card and SD I/O card can send only short responses, the

argument can vary according to the type of response: the software will distinguish the type of response according to the sent command. CE-ATA devices send only short responses.

26.9.5 SDIO command response register (SDIO_RESPCMD)

Address offset: 0x10

Reset value: 0x0000 0000

The SDIO_RESPCMD register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

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																										RESPCMD					
																										r	r	r	r	r	r

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

Bits 5:0 **RESPCMD:** Response command index

Read-only bit field. Contains the command index of the last command response received.

26.9.6 SDIO response 1..4 register (SDIO_RESPx)

Address offset: (0x10 + (4 × x)); x = 1..4

Reset value: 0x0000 0000

The SDIO_RESP1/2/3/4 registers contain the status of a card, which is part of the received response.

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
CARDSTATUSx																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:0 **CARDSTATUSx:** see [Table 133](#).

The Card Status size is 32 or 127 bits, depending on the response type.

Table 133. Response type and SDIO_RESPx registers

Register	Short response	Long response
SDIO_RESP1	Card Status[31:0]	Card Status [127:96]
SDIO_RESP2	Unused	Card Status [95:64]
SDIO_RESP3	Unused	Card Status [63:32]
SDIO_RESP4	Unused	Card Status [31:1]0b

The most significant bit of the card status is received first. The SDIO_RESP3 register LSB is always 0b.

26.9.7 SDIO data timer register (SDIO_DTIMER)

Address offset: 0x24

Reset value: 0x0000 0000

The SDIO_DTIMER register contains the data timeout period, in card bus clock periods.

A counter loads the value from the SDIO_DTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

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
DATATIME																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:0 **DATATIME:** Data timeout period

Data timeout period expressed in card bus clock periods.

Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

26.9.8 SDIO data length register (SDIO_DLEN)

Address offset: 0x28

Reset value: 0x0000 0000

The SDIO_DLEN register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

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		DATALENGTH																													
		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 31:25 Reserved, must be kept at reset value

Bits 24:0 **DATALENGTH:** Data length value

Number of data bytes to be transferred.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDIO_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDIO multibyte transfer the value in the data length register must be between 1 and 512.

26.9.9 SDIO data control register (SDIO_DCTRL)

Address offset: 0x2C

Reset value: 0x0000 0000

The SDIO_DCTRL register controls the data path state machine (DPSM).

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												SdioEN	RwMod	RwStop	RwStart	DBLOCKSIZE				Dmaen	DtMode	DtDir	DtEn								
												rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

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

Bit 11 **SDIOEN:** SD I/O enable functions

If this bit is set, the DPSM performs an SD I/O-card-specific operation.

Bit 10 **RWMOD:** Read wait mode

- 0: Read Wait control stopping SDIO_D2
- 1: Read Wait control using SDIO_CK

Bit 9 **RWSTOP:** Read wait stop

- 0: Read wait in progress if RWSTART bit is set
- 1: Enable for read wait stop if RWSTART bit is set

Bit 8 **RWSTART:** Read wait start

If this bit is set, read wait operation starts.

Bits 7:4 **DBLOCKSIZE:** Data block size

Define the data block length when the block data transfer mode is selected:

- 0000: (0 decimal) lock length = 2^0 = 1 byte
- 0001: (1 decimal) lock length = 2^1 = 2 bytes
- 0010: (2 decimal) lock length = 2^2 = 4 bytes
- 0011: (3 decimal) lock length = 2^3 = 8 bytes
- 0100: (4 decimal) lock length = 2^4 = 16 bytes
- 0101: (5 decimal) lock length = 2^5 = 32 bytes
- 0110: (6 decimal) lock length = 2^6 = 64 bytes
- 0111: (7 decimal) lock length = 2^7 = 128 bytes
- 1000: (8 decimal) lock length = 2^8 = 256 bytes
- 1001: (9 decimal) lock length = 2^9 = 512 bytes
- 1010: (10 decimal) lock length = 2^{10} = 1024 bytes
- 1011: (11 decimal) lock length = 2^{11} = 2048 bytes
- 1100: (12 decimal) lock length = 2^{12} = 4096 bytes
- 1101: (13 decimal) lock length = 2^{13} = 8192 bytes
- 1110: (14 decimal) lock length = 2^{14} = 16384 bytes
- 1111: (15 decimal) reserved

Bit 3 **DMAEN:** DMA enable bit

- 0: DMA disabled.
- 1: DMA enabled.

Bit 2 DTMODE: Data transfer mode selection 1: Stream or SDIO multibyte data transfer.

- 0: Block data transfer
 - 1: Stream or SDIO multibyte data transfer

Bit 1 **DTDIR**: Data transfer direction selection

- 0: From controller to card.
 - 1: From card to controller.

Bit 0 **DTEN**: Data transfer enabled bit

Data transfer starts if 1b is written to the DTEN bit. Depending on the direction bit, DTDIR, the DPSM moves to the Wait_S, Wait_R state or Readwait if RW Start is set immediately at the beginning of the transfer. It is not necessary to clear the enable bit after the end of a data transfer but the SDIO_DCTRL must be updated to enable a new data transfer

Note: After a data write, data cannot be written to this register for three SDIOCLK (48 MHz) clock periods plus two PCLK2 clock periods.

The meaning of the DTMODE bit changes according to the value of the SDIOEN bit. When SDIOEN=0 and DTMODE=1, the MultiMediaCard stream mode is enabled, and when SDIOEN=1 and DTMODE=1, the peripheral enables an SDIO multibyte transfer.

26.9.10 SDIO data counter register (SDIO_DCOUNT)

Address offset: 0x30

Reset value: 0x0000 0000

The SDIO_DCOUNT register loads the value from the data length register (see SDIO_DLEN) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the Idle state and the data status end flag, DATAEND, is set.

Bits 31:25 Reserved must be kept at reset value

Bits 24:0 **DATACOUNT**: Data count value

When this bit is read, the number of remaining data bytes to be transferred is returned. Write has no effect.

Note: This register should be read only when the data transfer is complete.

26.9.11 SDIO status register (SDIO_STA)

Address offset: 0x34

Reset value: 0x0000 0000

The SDIO_STA register is a read-only register. It contains two types of flag:

- Static flags (bits [23:22,10:0]): these bits remain asserted until they are cleared by writing to the SDIO Interrupt Clear register (see SDIO_ICR)
- Dynamic flags (bits [21:11]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

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																															
Res.					r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:24 Reserved, must be kept at reset value

Bit 23 **CEATAEND**: CE-ATA command completion signal received for CMD61

Bit 22 **SDIOIT**: SDIO interrupt received

Bit 21 **RXDAVL**: Data available in receive FIFO

Bit 20 **TXDAVL**: Data available in transmit FIFO

Bit 19 **RXFIFOE**: Receive FIFO empty

Bit 18 **TXFIFOE**: Transmit FIFO empty

When HW Flow Control is enabled, TXFIFOE signals becomes activated when the FIFO contains 2 words.

Bit 17 **RXFIFOF**: Receive FIFO full

When HW Flow Control is enabled, RXFIFOF signals becomes activated 2 words before the FIFO is full.

Bit 16 **TXFIFOF**: Transmit FIFO full

Bit 15 **RXFIFOHF**: Receive FIFO half full: there are at least 8 words in the FIFO

Bit 14 **TXFIFOHE**: Transmit FIFO half empty: at least 8 words can be written into the FIFO

Bit 13 **RXACT**: Data receive in progress

Bit 12 **TXACT**: Data transmit in progress

Bit 11 **CMDACT**: Command transfer in progress

Bit 10 **DBCKEND**: Data block sent/received (CRC check passed)

Bit 9 **STBITERR**: Start bit not detected on all data signals in wide bus mode

Bit 8 **DATAEND**: Data end (data counter, SDIDCOUNT, is zero)

Bit 7 **CMDSENT**: Command sent (no response required)

Bit 6 **CMDREND**: Command response received (CRC check passed)

Bit 5 **RXOVERR**: Received FIFO overrun error

- Bit 4 **TXUNDERR**: Transmit FIFO underrun error
- Bit 3 **DTIMEOUT**: Data timeout
- Bit 2 **CTIMEOUT**: Command response timeout
The Command TimeOut period has a fixed value of 64 SDIO_CK clock periods.
- Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)
- Bit 0 **CCRCFAIL**: Command response received (CRC check failed)

26.9.12 SDIO interrupt clear register (SDIO_ICR)

Address offset: 0x38

Reset value: 0x0000 0000

The SDIO_ICR register is a write-only register. Writing a bit with 1b clears the corresponding bit in the SDIO_STA Status 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		

Bits 31:24 Reserved, must be kept at reset value

Bit 23 **CEATAENDC**: CEATAEND flag clear bit

Set by software to clear the CEATAEND flag.
0: CEATAEND not cleared
1: CEATAEND cleared

Bit 22 **SDIOITC**: SDIOIT flag clear bit

Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared

Bits 21:11 Reserved, must be kept at reset value

Bit 10 **DBCKENDC**: DBCKEND flag clear bit

Set by software to clear the DBCKEND flag.
0: DBCKEND not cleared
1: DBCKEND cleared

Bit 9 **STBITERRC**: STBITERR flag clear bit

Set by software to clear the STBITERR flag.
0: STBITERR not cleared
1: STBITERR cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit

Set by software to clear the DATAEND flag.
0: DATAEND not cleared
1: DATAEND cleared

- Bit 7 **CMDSENTC:** CMDSENT flag clear bit
Set by software to clear the CMDSENT flag.
0: CMDSENT not cleared
1: CMDSENT cleared
- Bit 6 **CMDRENDC:** CMDREND flag clear bit
Set by software to clear the CMDREND flag.
0: CMDREND not cleared
1: CMDREND cleared
- Bit 5 **RXOVERRC:** RXOVERR flag clear bit
Set by software to clear the RXOVERR flag.
0: RXOVERR not cleared
1: RXOVERR cleared
- Bit 4 **TXUNDERRC:** TXUNDERR flag clear bit
Set by software to clear TXUNDERR flag.
0: TXUNDERR not cleared
1: TXUNDERR cleared
- Bit 3 **DTIMEOUTC:** DTIMEOUT flag clear bit
Set by software to clear the DTIMEOUT flag.
0: DTIMEOUT not cleared
1: DTIMEOUT cleared
- Bit 2 **CTIMEOUTC:** CTIMEOUT flag clear bit
Set by software to clear the CTIMEOUT flag.
0: CTIMEOUT not cleared
1: CTIMEOUT cleared
- Bit 1 **DCRCFAILC:** DCRCFAIL flag clear bit
Set by software to clear the DCRCFAIL flag.
0: DCRCFAIL not cleared
1: DCRCFAIL cleared
- Bit 0 **CCRCFAILC:** CCRCFAIL flag clear bit
Set by software to clear the CCRCFAIL flag.
0: CCRCFAIL not cleared
1: CCRCFAIL cleared

26.9.13 SDIO mask register (SDIO_MASK)

Address offset: 0x3C

Reset value: 0x0000 0000

The interrupt mask register determines which status flags generate an interrupt request by setting the corresponding bit to 1b.

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												CEATAENDIE	SDIOTIE	RXDAVIE	TXDAVIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHIE	TXFIFOHIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE

Bits 31:24 Reserved, must be kept at reset value

Bit 23 CEATAENDIE: CE-ATA command completion signal received interrupt enable

Set and cleared by software to enable/disable the interrupt generated when receiving the CE-ATA command completion signal.

- 0: CE-ATA command completion signal received interrupt disabled
- 1: CE-ATA command completion signal received interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable

Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.

0: SDIO Mode Interrupt Received interrupt disabled
1: SDIO Mode Interrupt Received interrupt enabled

Bit 21 RXDAVLIE: Data available in Rx FIFO interrupt enable

Set and cleared by software to enable/disable the interrupt generated by the presence of data available in Rx FIFO.

0: Data available in Rx FIFO interrupt disabled
1: Data available in Rx FIFO interrupt enabled

Bit 20 **TXDAVIE**: Data available in Tx FIFO interrupt enable

Set and cleared by software to enable/disable the interrupt generated by the presence of data available in Tx FIFO

0: Data available in Tx FIFO interrupt disabled
1: Data available in Tx FIFO interrupt enabled

Bit 19 **BXFIFOEIE**: Bx FIFO empty interrupt enable

Set and cleared by software to enable/disable interrupt caused by Rx FIFO empty.

0: Rx FIFO empty interrupt disabled
1: Rx FIFO empty interrupt enabled

Bit 18 TXFIFOEIE: Tx FIFO empty interrupt enable

Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.

0: Tx FIFO empty interrupt disabled
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOIE**: Rx FIFO full interrupt enable

Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.

0: Rx FIFO full interrupt disabled

1: Rx FIFO full interrupt enabled

- Bit 16 **TXFIFOIE:** Tx FIFO full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO full.
0: Tx FIFO full interrupt disabled
1: Tx FIFO full interrupt enabled
- Bit 15 **RXFIFOHFIE:** Rx FIFO half full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled
- Bit 14 **TXFIFOHEIE:** Tx FIFO half empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled
- Bit 13 **RXACTIE:** Data receive acting interrupt enable
Set and cleared by software to enable/disable interrupt caused by data being received (data receive acting).
0: Data receive acting interrupt disabled
1: Data receive acting interrupt enabled
- Bit 12 **TXACTIE:** Data transmit acting interrupt enable
Set and cleared by software to enable/disable interrupt caused by data being transferred (data transmit acting).
0: Data transmit acting interrupt disabled
1: Data transmit acting interrupt enabled
- Bit 11 **CMDACTIE:** Command acting interrupt enable
Set and cleared by software to enable/disable interrupt caused by a command being transferred (command acting).
0: Command acting interrupt disabled
1: Command acting interrupt enabled
- Bit 10 **DBCKENDIE:** Data block end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled
- Bit 9 **STBITERRIE:** Start bit error interrupt enable
Set and cleared by software to enable/disable interrupt caused by start bit error.
0: Start bit error interrupt disabled
1: Start bit error interrupt enabled
- Bit 8 **DATAENDIE:** Data end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled
- Bit 7 **CMDSENTIE:** Command sent interrupt enable
Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

- Bit 6 **CMDRENDIE:** Command response received interrupt enable
Set and cleared by software to enable/disable interrupt caused by receiving command response.
0: Command response received interrupt disabled
1: command Response Received interrupt enabled

Bit 5 **RXOVERRIE:** Rx FIFO overrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUNDERRIE:** Tx FIFO underrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **DTIMEOUTIE:** Data timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CTIMEOUTIE:** Command timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRCFAILIE:** Data CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRCFAILIE:** Command CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled

26.9.14 SDIO FIFO counter register (SDIO_FIFOCNT)

Address offset: 0x48

Reset value: 0x0000 0000

The SDIO_FIFOCNT register contains the remaining number of words to be written to or read from the FIFO. The FIFO counter loads the value from the data length register (see SDIO_DLEN) when the data transfer enable bit, DTEN, is set in the data control register (SDIO_DCTRL register) and the DPSM is at the Idle state. If the data length is not word-aligned (multiple of 4), the remaining 1 to 3 bytes are regarded as a word.

Bits 31:24 Reserved, must be kept at reset value

Bits 23:0 **FIFOCOUNT:** Remaining number of words to be written to or read from the FIFO.

26.9.15 SDIO data FIFO register (SDIO_FIFO)

Address offset: 0x80

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This allows the CPU to use its load and store multiple operands to read from/write to the FIFO.

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
FIFOData																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

bits 31:0 **FIFOData**: Receive and transmit FIFO data

The FIFO data occupies 32 entries of 32-bit words, from address:
SDIO base + 0x080 to SDIO base + 0xFC.

26.9.16 SDIO register map

The following table summarizes the SDIO registers.

Table 134. SDIO 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	SDIO_POWER																																
0x04	SDIO_CLKCR																																
0x08	SDIO_ARG																																
0x0C	SDIO_CMD																																
0x10	SDIO_RESPCMD																																
0x14	SDIO_RESP1																																
0x18	SDIO_RESP2																																
0x1C	SDIO_RESP3																																
0x20	SDIO_RESP4																																
0x24	SDIO_DTIMER																																
0x28	SDIO_DLEN																																
0x2C	SDIO_DCTRL																																
0x30	SDIO_DCOUNT																																
0x34	SDIO_STA																																

Table 134. SDIO register map (continued)

Refer to [Table 1 on page 50](#) for the register boundary addresses.

31 Flexible static memory controller (FSMC)

31.1 FSMC main features

The FSMC block is able to interface with synchronous and asynchronous memories and 16-bit PC memory cards. Its main purpose is to:

- Translate the AHB transactions into the appropriate external device protocol
- Meet the access timing requirements of the external devices

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

The FSMC has the following main features:

- Interfaces with static memory-mapped devices including:
 - Static random access memory (SRAM)
 - Read-only memory (ROM)
 - NOR Flash memory/OneNAND Flash memory
 - PSRAM (4 memory banks)
- Two banks of NAND Flash with ECC hardware that checks up to 8 Kbytes of data
- 16-bit PC Card compatible devices
- Supports burst mode access to synchronous devices (NOR Flash and PSRAM)
- 8- or 16-bit wide databus
- Independent chip select control for each memory bank
- Independent configuration for each memory bank
- Programmable timings to support a wide range of devices, in particular:
 - Programmable wait states (up to 15)
 - Programmable bus turnaround cycles (up to 15)
 - Programmable output enable and write enable delays (up to 15)
 - Independent read and write timings and protocol, so as to support the widest variety of memories and timings
- Write enable and byte lane select outputs for use with PSRAM and SRAM devices
- Translation of 32-bit wide AHB transactions into consecutive 16-bit or 8-bit accesses to external 16-bit or 8-bit devices
- A Write FIFO, 2 words long, each word is 32 bits wide, only stores data and not the address. Therefore, this FIFO only buffers AHB write burst transactions. This makes it possible to write to slow memories and free the AHB quickly for other operations. Only one burst at a time is buffered: if a new AHB burst or single transaction occurs while an operation is in progress, the FIFO is drained. The FSMC will insert wait states until the current memory access is complete).
- External asynchronous wait control

The FSMC registers that define the external device type and associated characteristics are usually set at boot time and do not change until the next reset or power-up. However, it is possible to change the settings at any time.

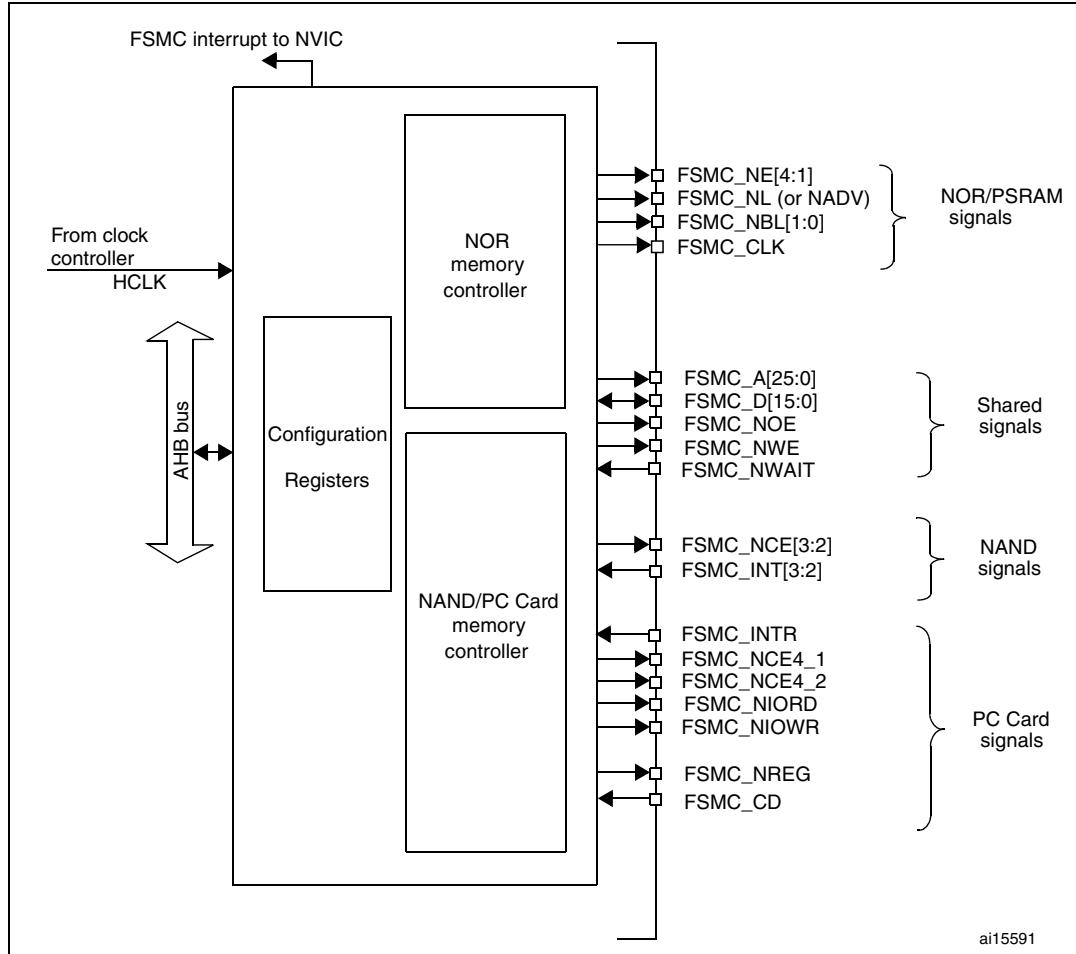
31.2 Block diagram

The FSMC consists of four main blocks:

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

The block diagram is shown in [Figure 388](#).

Figure 388. FSMC block diagram



31.3 AHB interface

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

AHB transactions are translated into the external device protocol. In particular, if the selected external memory is 16 or 8 bits wide, 32-bit wide transactions on the AHB are split into consecutive 16- or 8-bit accesses. The Chip Select toggles for each access.

The FSMC generates an AHB error in the following conditions:

- When reading or writing to an FSMC bank which is not enabled
- When reading or writing to the NOR Flash bank while the FACCEN bit is reset in the FSMC_BCRx register.
- When reading or writing to the PC Card banks while the input pin FSMC_CD (Card Presence Detection) is low.

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

- If it is the Cortex™-M4F CPU, a hard fault interrupt is generated
- If it is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

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

31.3.1 Supported memories and transactions

General transaction rules

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

Therefore, some simple transaction rules must be followed:

- AHB transaction size and memory data size are equal
There is no issue in this case.
- AHB transaction size is greater than the memory size
In this case, the FSMC splits the AHB transaction into smaller consecutive memory accesses in order to meet the external data width.
- AHB transaction size is smaller than the memory size
Asynchronous transfers may or not be consistent depending on the type of external device.
 - Asynchronous accesses to devices that have the byte select feature (SRAM, ROM, PSRAM).
 - a) FSMC allows write transactions accessing the right data through its byte lanes NBL[1:0]
 - b) Read transactions are allowed (the controller reads the entire memory word and uses the needed byte only). The NBL[1:0] are always kept low during read transactions.
 - Asynchronous accesses to devices that do not have the byte select feature (NOR and NAND Flash 16-bit).
This situation occurs when a byte access is requested to a 16-bit wide Flash memory. Clearly, the device cannot be accessed in byte mode (only 16-bit words can be read from/written to the Flash memory) therefore:
 - a) Write transactions are not allowed
 - b) Read transactions are allowed (the controller reads the entire 16-bit memory word and uses the needed byte only).

Configuration registers

The FSMC can be configured using a register set. See [Section 31.5.6](#), for a detailed description of the NOR Flash/PSRAM controller registers. See [Section 31.6.8](#), for a detailed description of the NAND Flash/PC Card registers.

31.4 External device address mapping

From the FSMC point of view, the external memory is divided into 4 fixed-size banks of 256 Mbytes each (Refer to [Figure 389](#)):

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

For each bank the type of memory to be used is user-defined in the Configuration register.

Figure 389. FSMC memory banks

Address	Banks	Supported memory type
6000 0000h 6FFF FFFFh	Bank 1 4 × 64 MB	NOR / PSRAM
7000 0000h 7FFF FFFFh	Bank 2 4 × 64 MB	NAND Flash
8000 0000h 8FFF FFFFh	Bank 3 4 × 64 MB	
9000 0000h 9FFF FFFFh	Bank 4 4 × 64 MB	PC Card

ai14719

31.4.1 NOR/PSRAM address mapping

HADDR[27:26] bits are used to select one of the four memory banks as shown in [Table 163](#).

Table 163. NOR/PSRAM bank selection

HADDR[27:26] ⁽¹⁾	Selected bank
00	Bank 1 NOR/PSRAM 1
01	Bank 1 NOR/PSRAM 2

Table 163. NOR/PSRAM bank selection (continued)

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

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

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

Table 164. External memory address

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

1. In case of a 16-bit external memory width, the FSMC will internally use HADDR[25:1] to generate the address for external memory FSMC_A[24:0].
Whatever the external memory width (16-bit or 8-bit), FSMC_A[0] should be connected to external memory address A[0].

Wrap support for NOR Flash/PSRAM

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

31.4.2 NAND/PC Card address mapping

In this case, three banks are available, each of them divided into memory spaces as indicated in [Table 165](#).

Table 165. Memory mapping and timing registers

Start address	End address	FSMC Bank	Memory space	Timing register
0x9C00 0000	0x9FFF FFFF	Bank 4 - PC card	I/O	FSMC_PIO4 (0xB0)
0x9800 0000	0x9BFF FFFF		Attribute	FSMC_PATT4 (0xAC)
0x9000 0000	0x93FF FFFF		Common	FSMC_PMEM4 (0xA8)
0x8800 0000	0x8BFF FFFF	Bank 3 - NAND Flash	Attribute	FSMC_PATT3 (0x8C)
0x8000 0000	0x83FF FFFF		Common	FSMC_PMEM3 (0x88)
0x7800 0000	0x7BFF FFFF	Bank 2- NAND Flash	Attribute	FSMC_PATT2 (0x6C)
0x7000 0000	0x73FF FFFF		Common	FSMC_PMEM2 (0x68)

For NAND Flash memory, the common and attribute memory spaces are subdivided into three sections (see in [Table 166](#) 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 166. NAND bank selections

Section name	HADDR[17:16]	Address range
Address section	1X	0x020000-0x03FFFF
Command section	01	0x010000-0x01FFFF
Data section	00	0x000000-0x0FFFF

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

- **To send 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 writes to the address section are needed to specify the full address.
- **To read or write data:** the software reads or writes the data value from or 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.

31.5 NOR Flash/PSRAM controller

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

- Asynchronous SRAM and ROM
 - 8-bit
 - 16-bit
 - 32-bit
- PSRAM (Cellular RAM)
 - Asynchronous mode
 - Burst mode
 - Multiplexed or nonmultiplexed
- NOR Flash
 - Asynchronous mode or burst mode
 - Multiplexed or nonmultiplexed

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

For synchronous accesses, the FSMC issues the clock (CLK) to the selected external device. This clock is a submultiple of the HCLK clock. The size of each bank is fixed and equal to 64 Mbytes.

Each bank is configured by means of dedicated registers (see [Section 31.5.6](#)).

The programmable memory parameters include access timings (see [Table 167](#)) and support for wait management (for PSRAM and NOR Flash accessed in burst mode).

Table 167. 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
Burst turn	Duration of the bus turnaround phase	Asynchronous and synchronous read	AHB clock cycle (HCLK)	0	15
Clock divide ratio	Number of AHB clock cycles (HCLK) to build one memory clock cycle (CLK)	Synchronous	AHB clock cycle (HCLK)	1	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (CLK)	2	17

31.5.1 External memory interface signals

Table 168, *Table 169* and *Table 170* list the signals that are typically used to interface NOR Flash, SRAM and PSRAM.

Note: Prefix “N”. specifies the associated signal as active low.

NOR Flash, nonmultiplexed I/Os

Table 168. Nonmultiplexed I/O NOR Flash

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

NOR Flash memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

NOR Flash, multiplexed I/Os

Table 169. Multiplexed I/O NOR Flash

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

NOR-Flash memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

PSRAM/SRAM, nonmultiplexed I/Os

Table 170. Nonmultiplexed I/Os PSRAM/SRAM

FSMC signal name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous burst)
A[25:0]	O	Address bus
D[15:0]	I/O	Data bidirectional bus
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FSMC
NBL[1]	O	Upper byte enable (memory signal name: NUB)
NBL[0]	O	Lower byte enable (memory signal name: NLB)

PSRAM memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

PSRAM, multiplexed I/Os

Table 171. Multiplexed I/O PSRAM

FSMC signal name	I/O	Function
CLK	O	Clock (for synchronous burst)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NE[x]	O	Chip select, $x = 1..4$ (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FSMC
NBL[1]	O	Upper byte enable (memory signal name: NUB)
NBL[0]	O	Lower byte enable (memory signal name: NLB)

PSRAM memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

31.5.2 Supported memories and transactions

Table 172 below displays an example of the supported devices, access modes and transactions when the memory data bus is 16-bit for NOR, PSRAM and SRAM. Transactions not allowed (or not supported) by the FSMC in this example appear in gray.

Table 172. NOR Flash/PSRAM supported memories and transactions

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

Table 172. NOR Flash/PSRAM supported memories and transactions (continued)

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

31.5.3 General timing rules

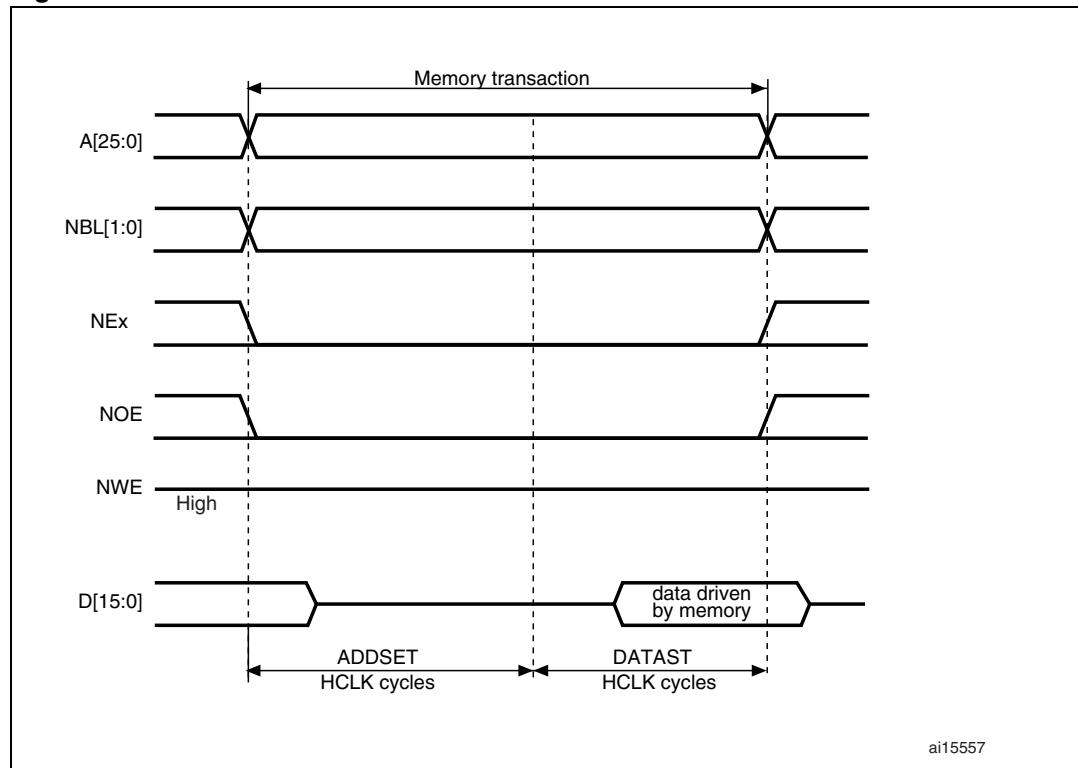
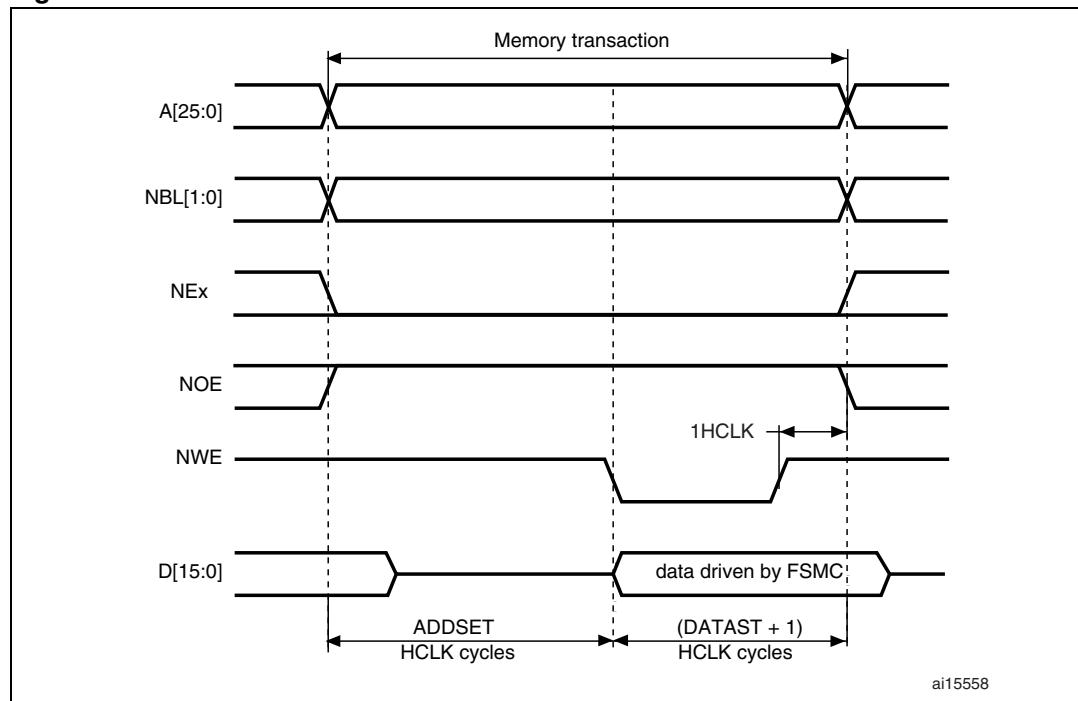
Signals synchronization

- All controller output signals change on the rising edge of the internal clock (HCLK)
- In synchronous read and write mode, the output data changes on the falling edge of the memory clock (FSMC_CLK).

31.5.4 NOR Flash/PSRAM controller asynchronous transactions

Asynchronous static memories (NOR Flash, SRAM)

- Signals are synchronized by the internal clock HCLK. This clock is not issued to the memory
- The FSMC always samples the data before de-asserting the chip select signal NE. This guarantees that the memory data-hold timing constraint is met (chip enable high to data transition, usually 0 ns min.)
- When extended mode is set, it is possible to mix modes A, B, C and D in read and write (it is for instance possible to read in mode A and write in mode B).

Mode 1 - SRAM/CRAM**Figure 390. Mode1 read accesses****Figure 391. Mode1 write accesses**

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

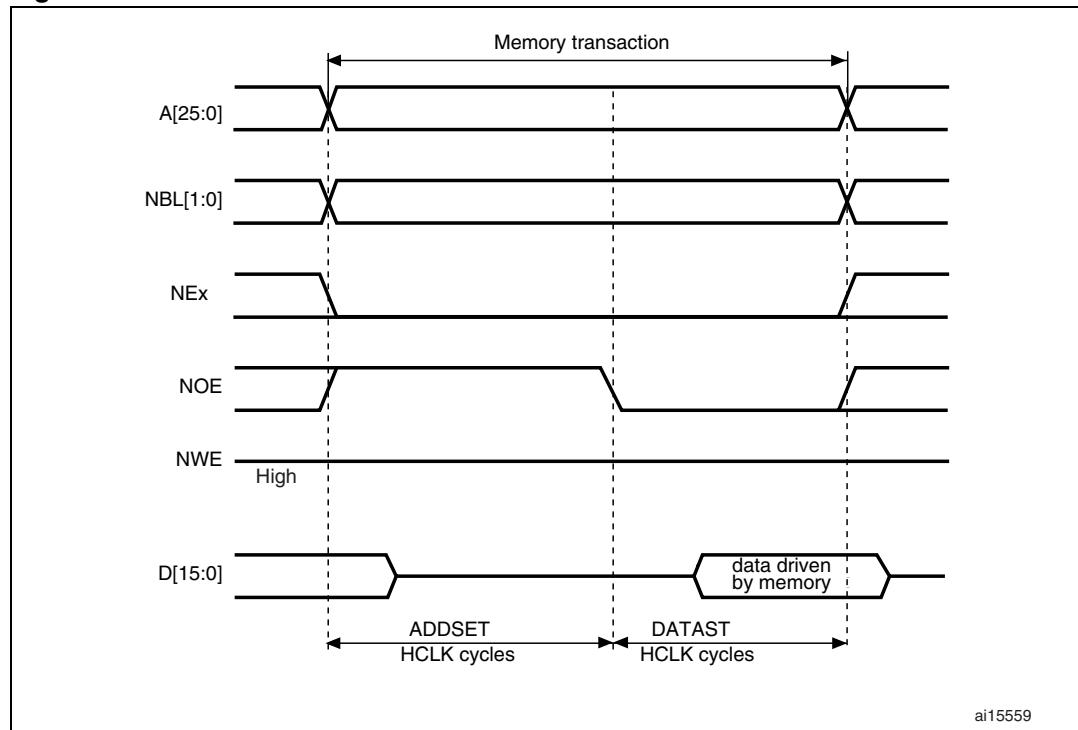
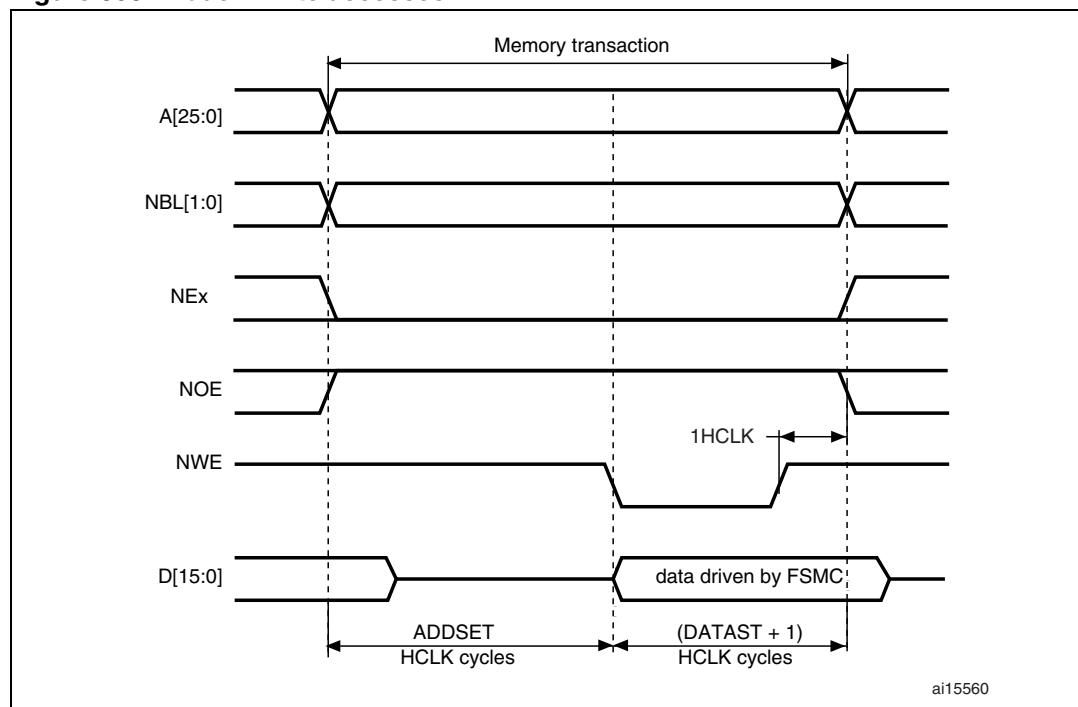
The one HCLK cycle at the end of the write transaction helps guarantee the address and data hold time after the NWE rising edge. Due to the presence of this one HCLK cycle, the DATAST value must be greater than zero (DATAST > 0).

Table 173. FSMC_BCRx bit fields

Bit number	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	-
5-4	MWID	As needed
3-2	MTYP	As needed, exclude 10 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 174. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31-20		0x0000
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		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 0.

Mode A - SRAM/PSRAM (CRAM) OE toggling**Figure 392. ModeA read accesses****Figure 393. ModeA write accesses**

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

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

Table 175. FSMC_BCRx bit fields

Bit number	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	-
5-4	MWID	As needed
3-2	MTYP	As needed, exclude 10 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 176. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x0
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) in read.
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in read. Minimum value for ADDSET is 1.

Table 177. FSMC_BWTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x0
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).

Table 177. FSMC_BWTRx bit fields (continued)

Bit number	Bit name	Value to set
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles) in write.
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in write Minimum value for ADDSET is 1.

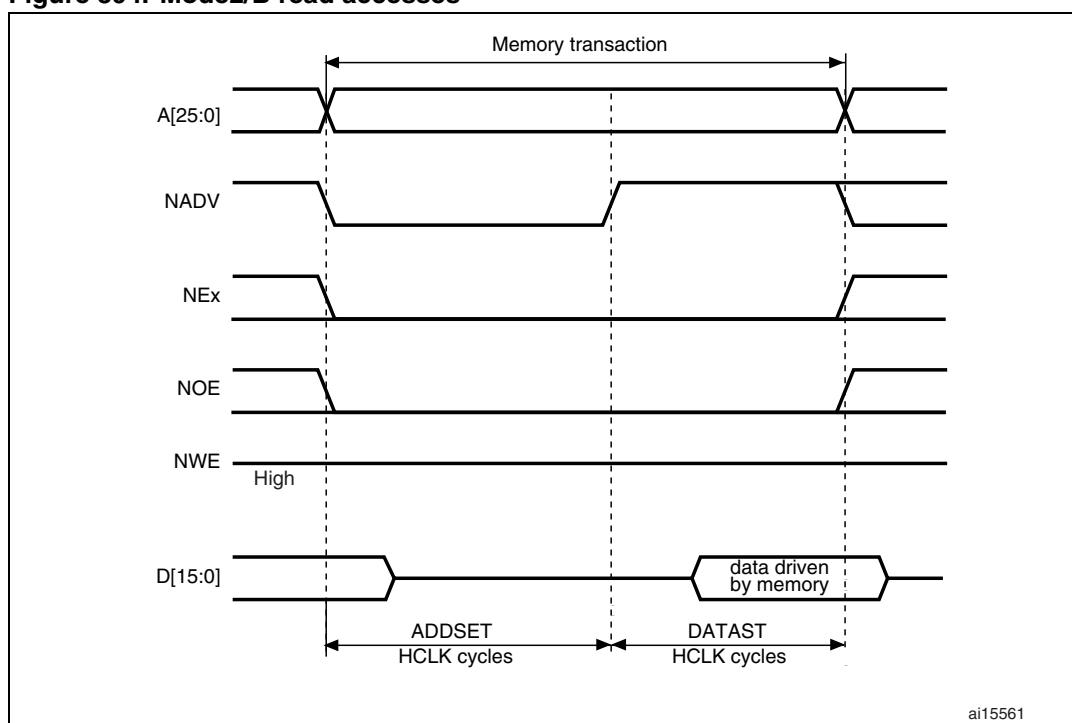
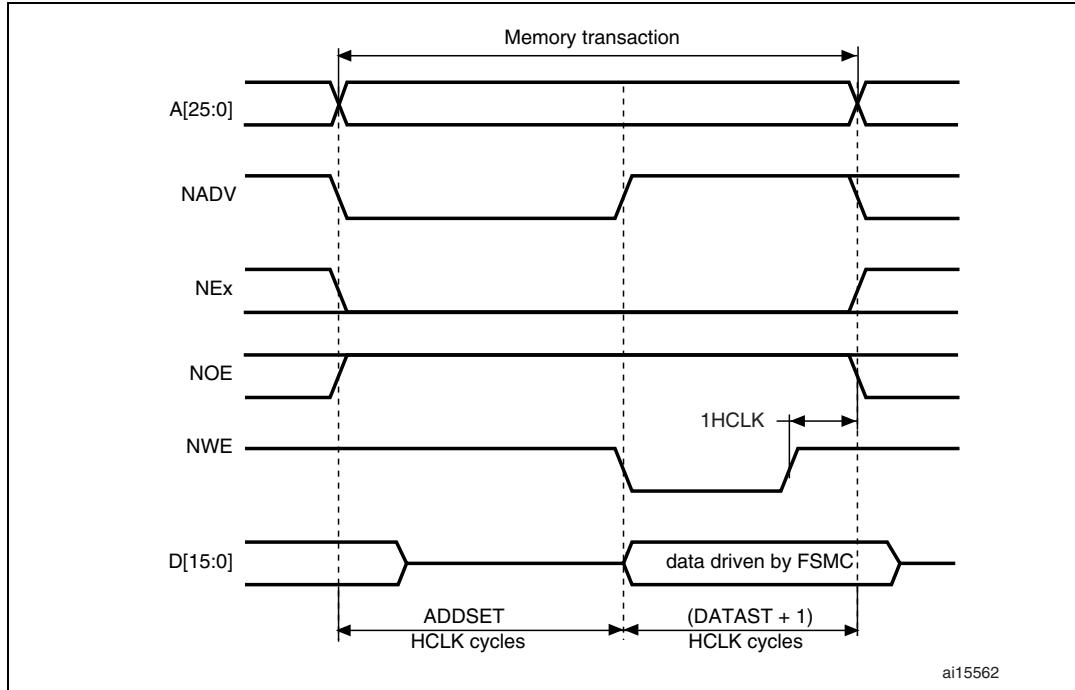
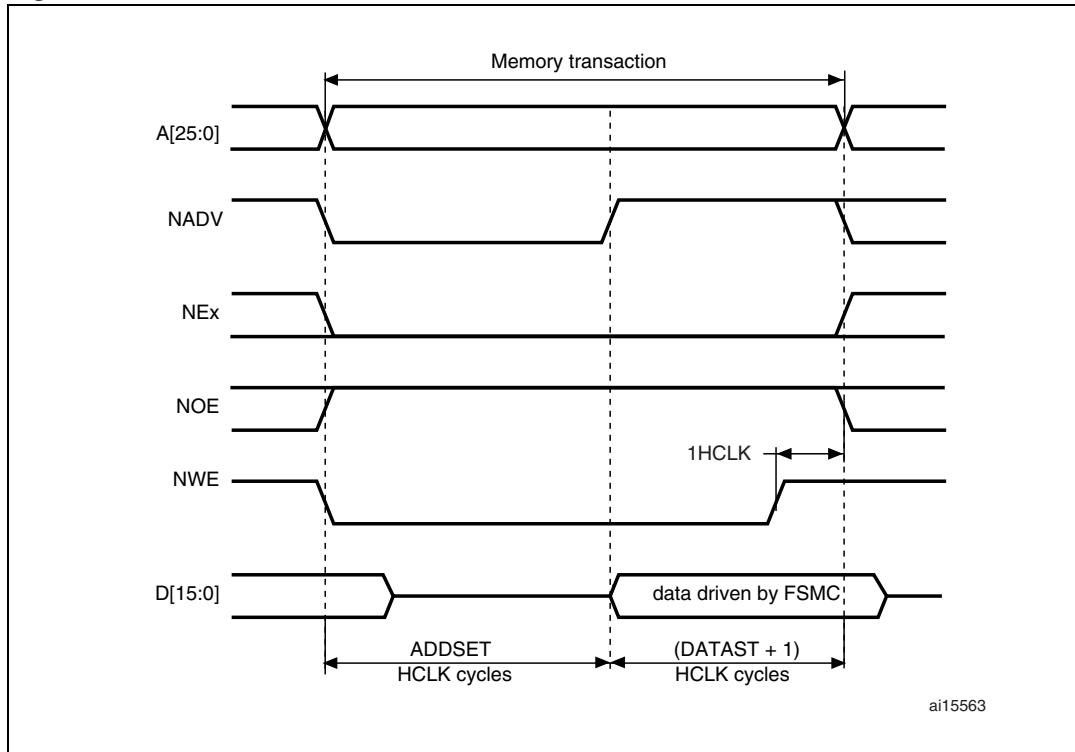
Mode 2/B - NOR Flash**Figure 394. Mode2/B read accesses**

Figure 395. Mode2 write accesses**Figure 396. ModeB write accesses**

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

Table 178. FSMC_BCRx bit fields

Bit number	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	10 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 179. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x1 if extended mode is set
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the access second phase (DATAST HCLK cycles) in read.
7-4		0x0
3-0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) in read. Minimum value for ADDSET is 1.

Table 180. FSMC_BWTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x1 if extended mode is set
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the access second phase (DATAST+1 HCLK cycles) in write.

Table 180. FSMC_BWTRx bit fields (continued)

Bit number	Bit name	Value to set
7-4		0x0
3-0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) in write. Minimum value for ADDSET is 1.

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

Mode C - NOR Flash - OE toggling

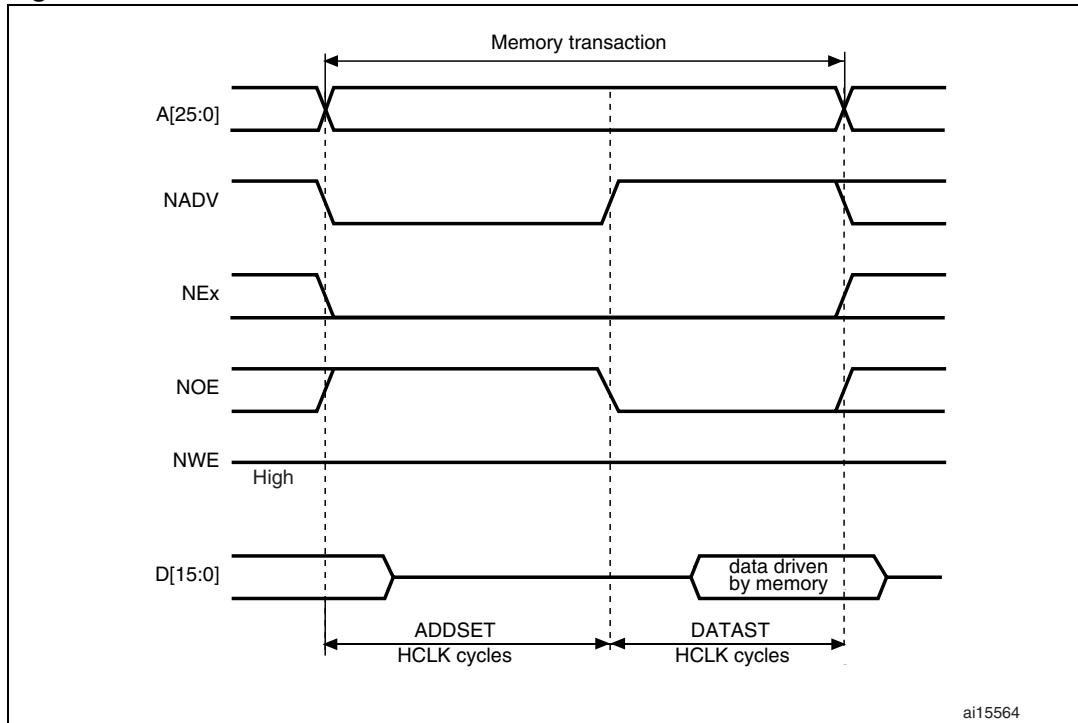
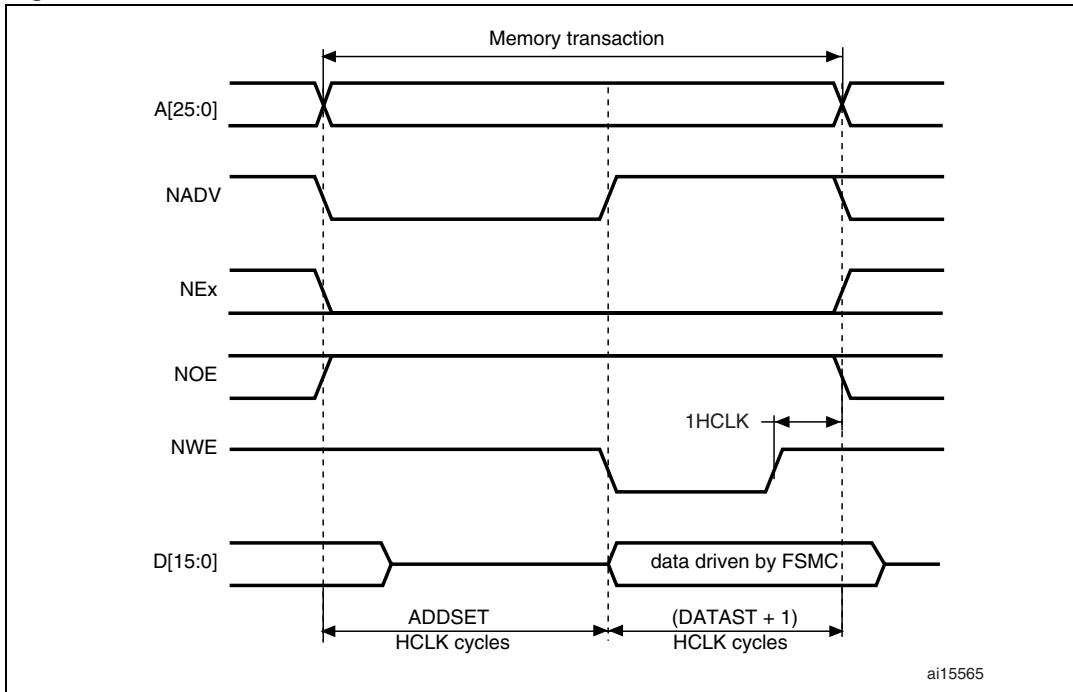
Figure 397. ModeC read accesses

Figure 398. ModeC write accesses

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

Table 181. FSMC_BCRx bit fields

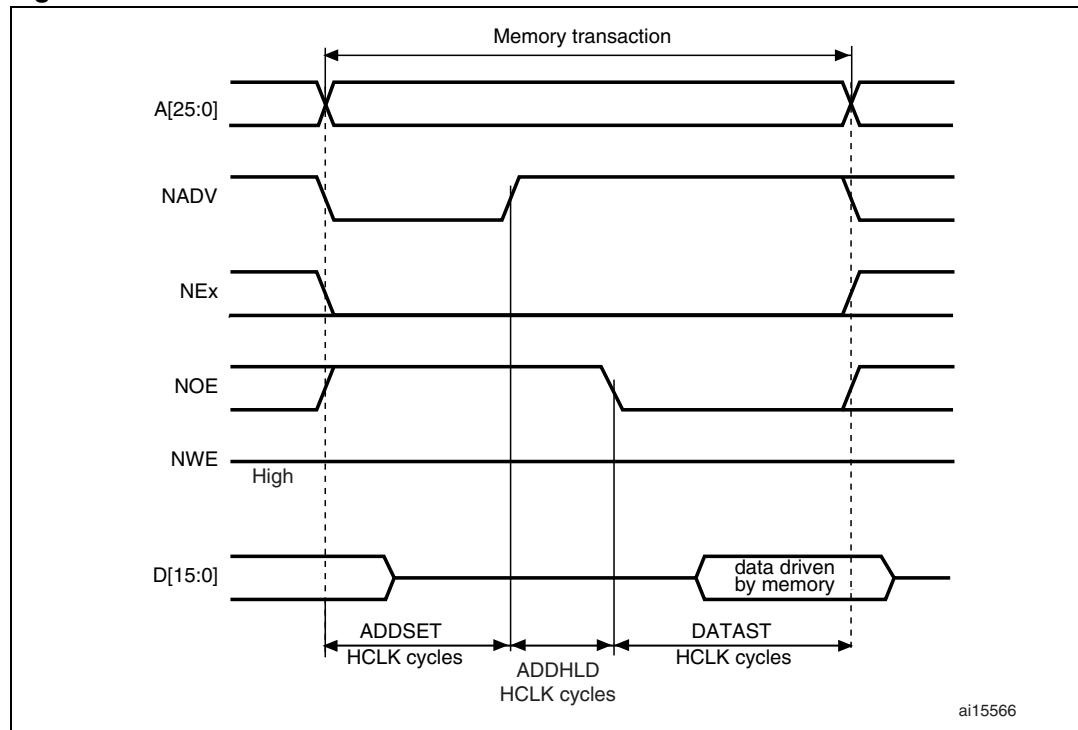
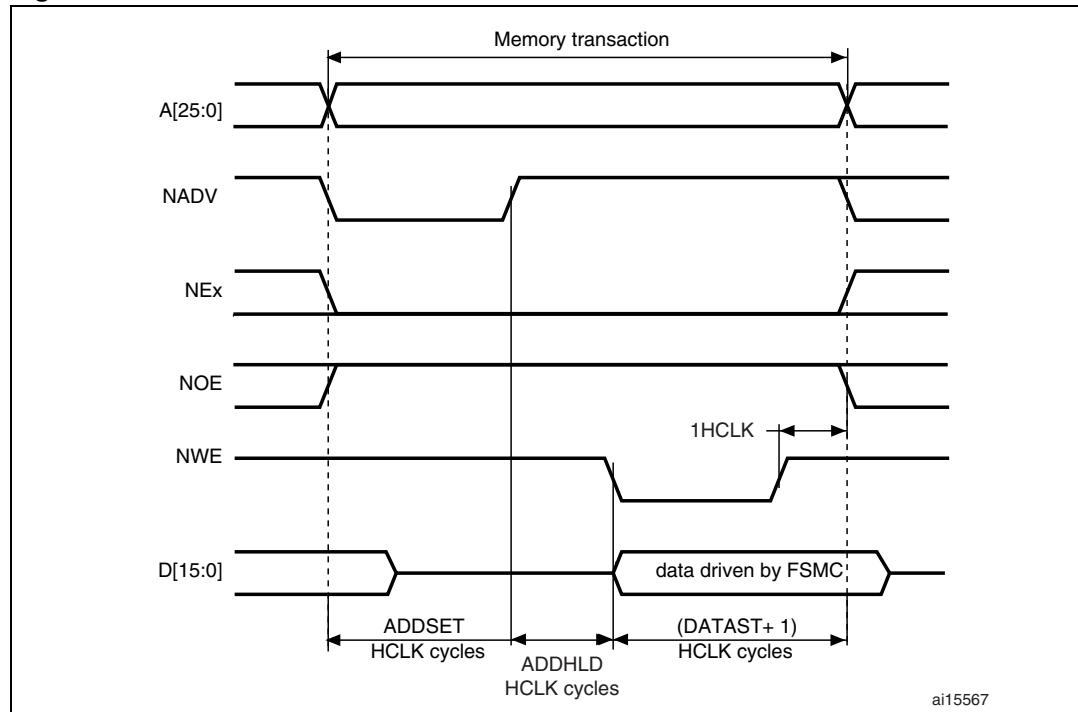
Bit No.	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	1
5-4	MWID	As needed
3-2	MTYP	0x02 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 182. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) in read.
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in read. Minimum value for ADDSET is 1.

Table 183. FSMC_BWTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
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) in write.
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in write. Minimum value for ADDSET is 1.

Mode D - asynchronous access with extended address**Figure 399. ModeD read accesses****Figure 400. ModeD write accesses**

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

Table 184. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 185. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) in read.
7-4	ADDHLD	Duration of the middle phase of the read access (ADDHLD HCLK cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in read. Minimum value for ADDSET is 1.

Table 186. FSMC_BWTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
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) in write.

Table 186. FSMC_BWTRx bit fields (continued)

Bit No.	Bit name	Value to set
7-4	ADDHLD	Duration of the middle phase of the write access (ADDHLD HCLK cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in write. Minimum value for ADDSET is 1.

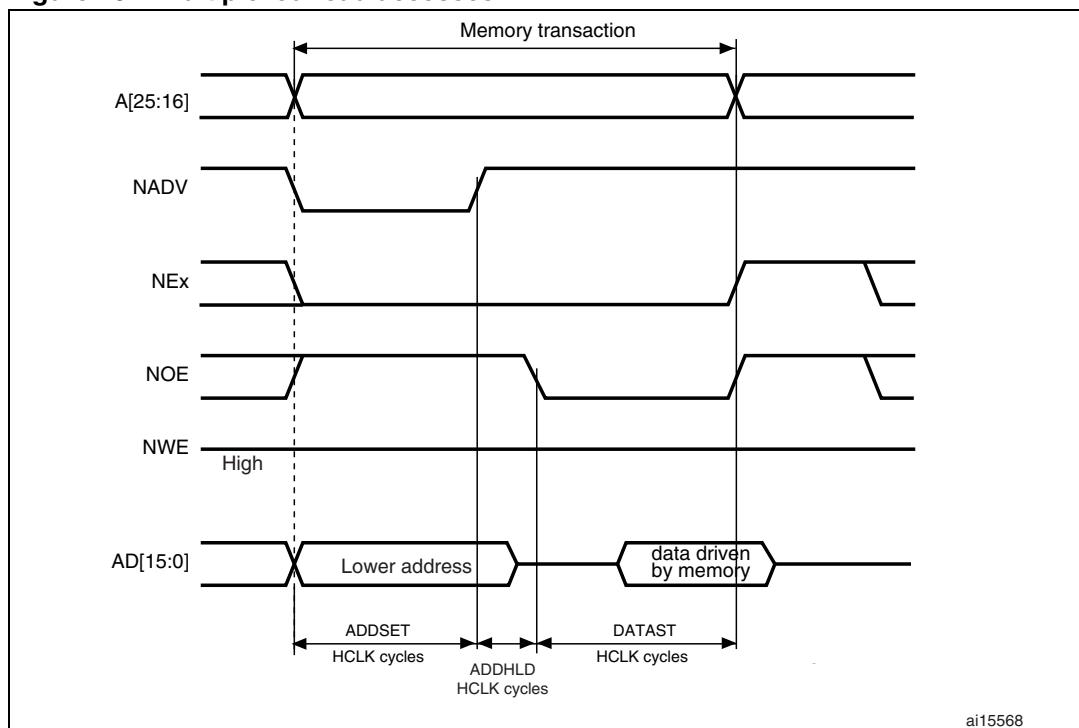
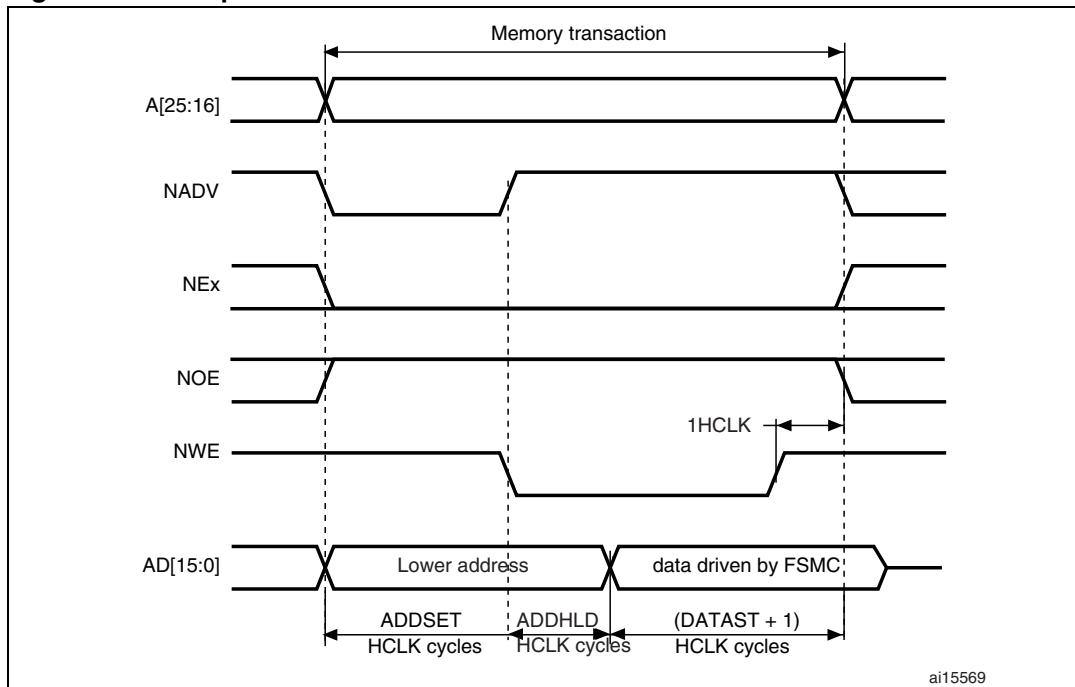
Mode muxed - asynchronous access muxed NOR Flash**Figure 401. Multiplexed read accesses**

Figure 402. Multiplexed write accesses

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

Table 187. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	0x2 (NOR)
1	MUXEN	0x1
0	MBKEN	0x1

Table 188. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-20		0x0000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)

Table 188. FSMC_BTRx bit fields (continued)

Bit No.	Bit name	Value to set
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles for read accesses and DATAST+1 HCLK cycles for write accesses).
7-4	ADDHLD	Duration of the middle phase of the access (ADDHLD HCLK cycles).
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 1.

WAIT management in asynchronous accesses

If the asynchronous memory asserts a WAIT signal to advise that it's not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FSMC_BCRx register.

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

The data phase must be programmed so that WAIT can be detected 4 HCLK cycles before the data sampling. The following cases must be considered:

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

$$\text{data_setup phase} \geq 4 * \text{HCLK} + \text{max_wait_assertion_time}$$
2. Memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):

$$\text{if } \text{max_wait_assertion_time} > (\text{address_phase} + \text{hold_phase})$$

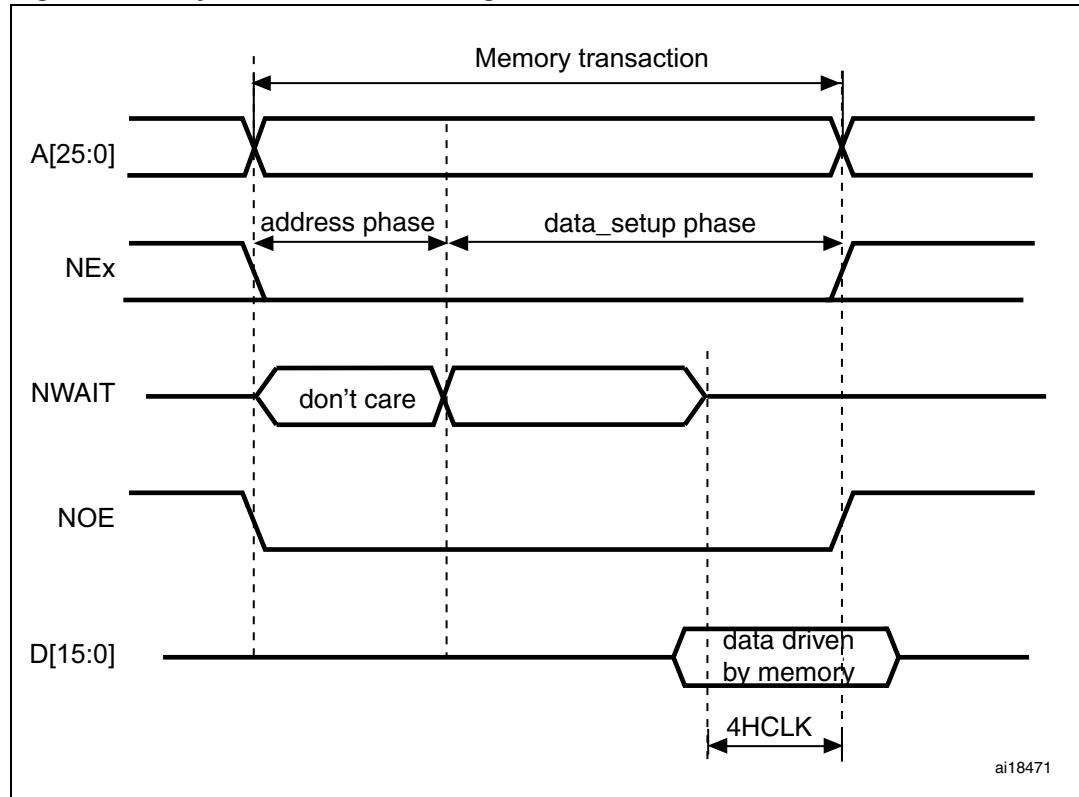
$$\text{data_setup phase} \geq 4 * \text{HCLK} + (\text{max_wait_assertion_time} - \text{address_phase} - \text{hold_phase})$$

$$\text{otherwise}$$

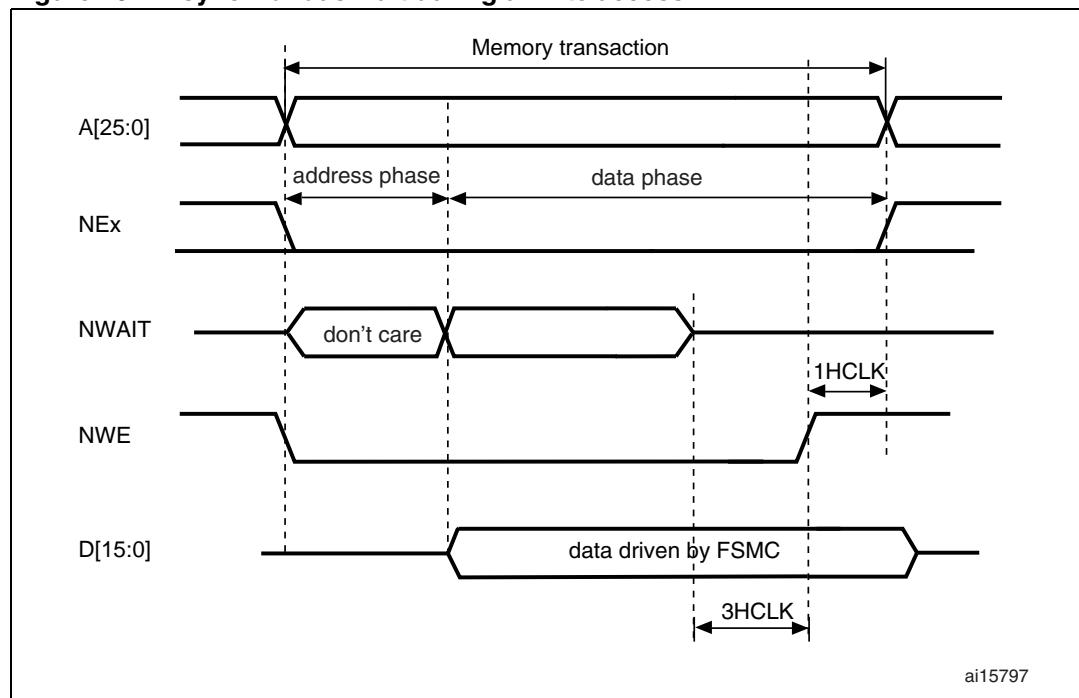
$$\text{data_setup phase} \geq 4 * \text{HCLK}$$

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

The [Figure 403](#) and [Figure 404](#) show the number of HCLK clock cycles that memory access is extended after WAIT is removed by the asynchronous memory (independently of the above cases).

Figure 403. Asynchronous wait during a read access

ai18471

Figure 404. Asynchronous wait during a write access

ai15797

31.5.5 Synchronous burst transactions

The memory clock, CLK, is a submultiple of HCLK according to the value of parameter CLKDIV.

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

Data latency versus NOR Flash latency

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

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

- NOR Flash latency = DATLAT + 2
- NOR Flash latency = DATLAT + 3

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

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

Single-burst transfer

When the selected bank is configured in synchronous burst mode, if an AHB single-burst transaction is requested, the FSMC performs a burst transaction of length 1 (if the AHB transfer is 16-bit), or length 2 (if the AHB transfer is 32-bit) and de-assert the chip select signal when the last data is strobed.

Clearly, such a transfer is not the most efficient in terms of cycles (compared to an asynchronous read). Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

Wait management

For synchronous burst NOR Flash, NWAIT is evaluated after the programmed latency period, (DATALAT+2) CLK clock cycles.

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

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

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

There are two timing configurations for the NOR Flash NWAIT signal in burst mode:

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

These two NOR Flash wait state configurations are supported by the FSMC, individually for each chip select, thanks to the WAITCFG bit in the FSMC_BCRx registers ($x = 0..3$).

Figure 405. Wait configurations

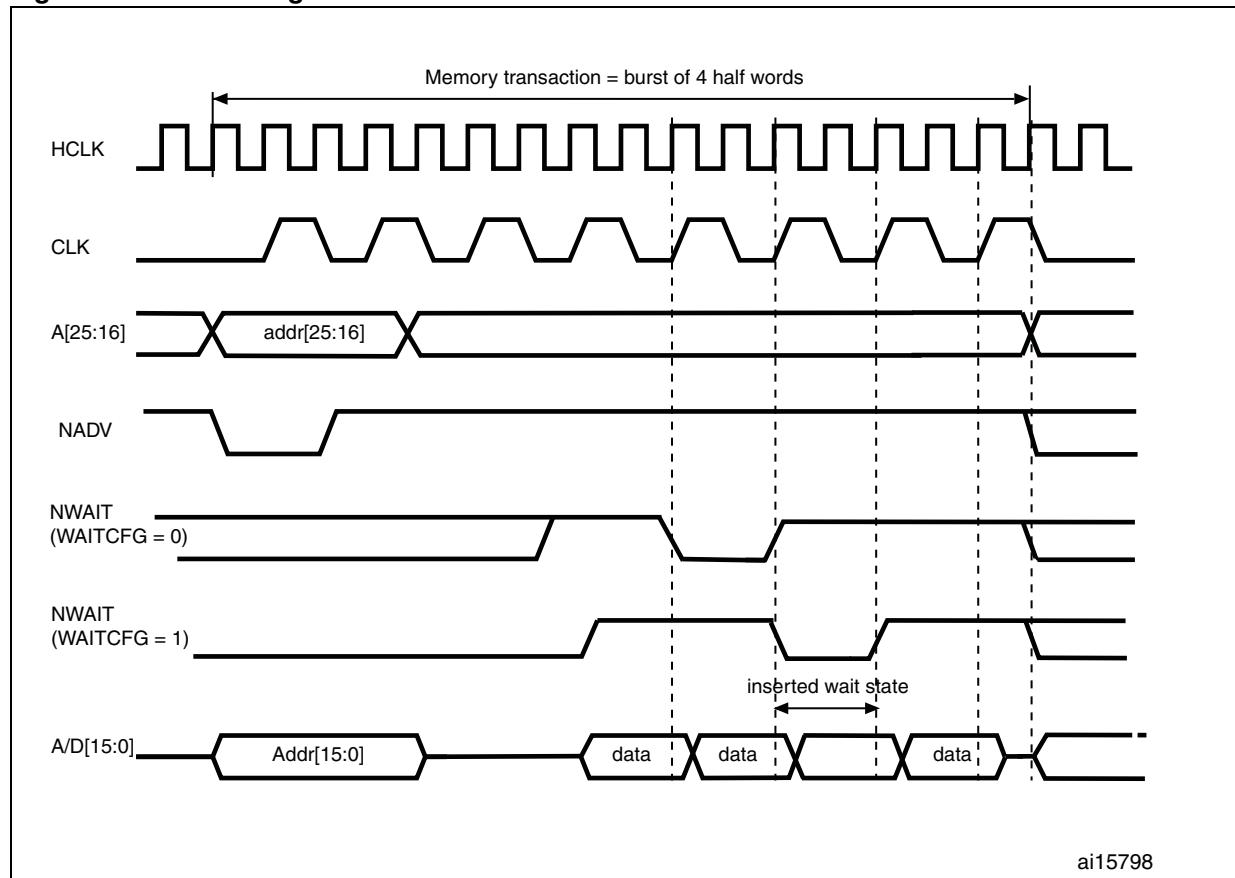
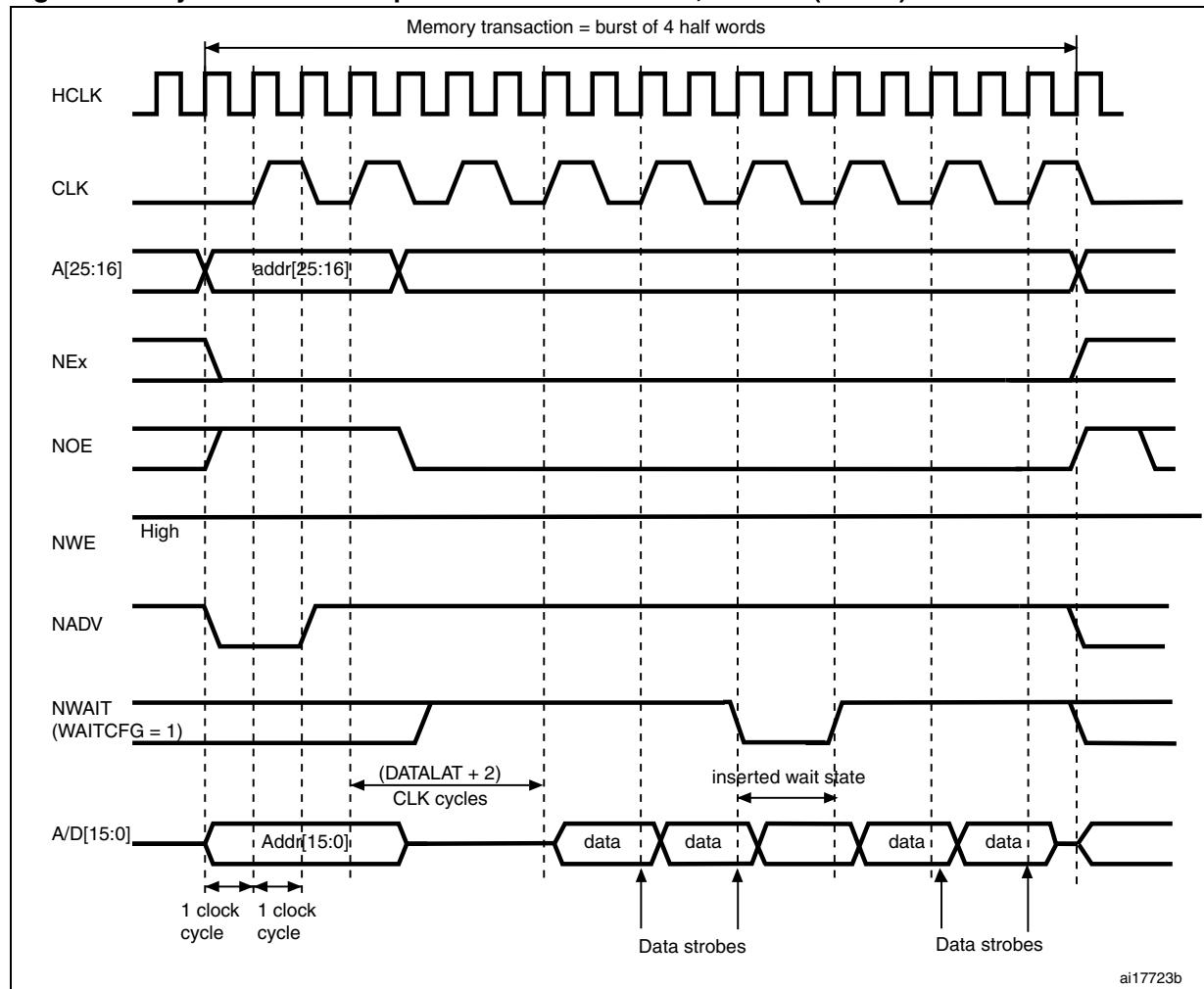


Figure 406. Synchronous multiplexed read mode - NOR, PSRAM (CRAM)



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

Table 189. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-20		0x0000
19	CBURSTRW	No effect on synchronous read
18-15		0x0
14	EXTMOD	0x0
13	WAITEN	When high, the first data after latency period is taken as always valid, regardless of the wait from memory value
12	WREN	no effect on synchronous read
11	WAITCFG	to be set according to memory
10	WRAPMOD	no effect
9	WAITPOL	to be set according to memory

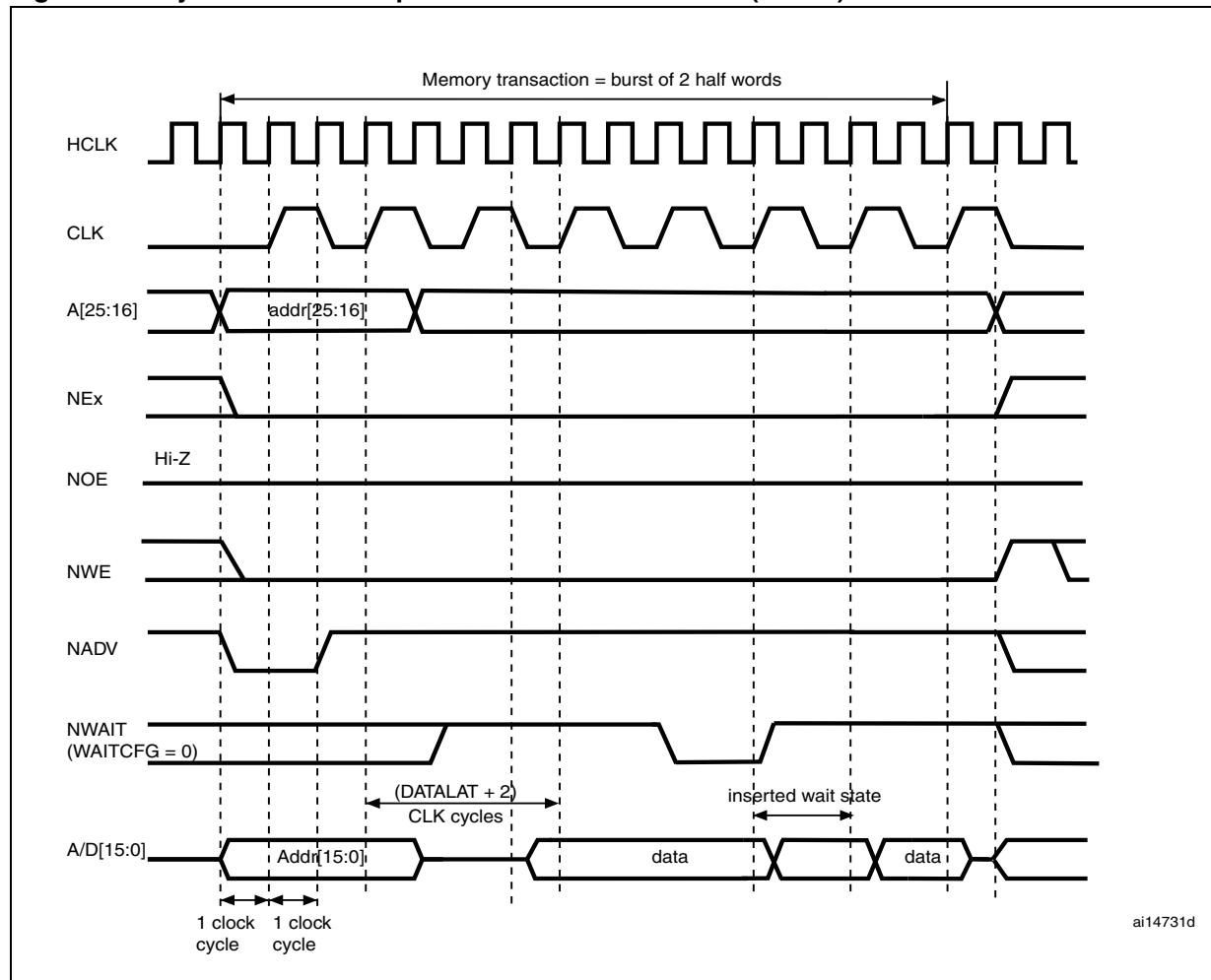
Table 189. FSMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
8	BURSTEN	0x1
7	FWPRLVL	Set to protect memory from accidental write access
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 190. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK 0x1 to get CLK = 2 × HCLK ..
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	no effect
7-4	ADDHLD	no effect
3-0	ADDSET	no effect

Figure 407. Synchronous multiplexed write mode - PSRAM (CRAM)



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

Table 191. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-20		0x0000
19	CBURSTRW	0x1
18-15		0x0
14	EXTMOD	0x0
13	WAITEN	When high, the first data after latency period is taken as always valid, regardless of the wait from memory value
12	WREN	no effect on synchronous read
11	WAITCFG	0x0
10	WRAPMOD	no effect
9	WAITPOL	to be set according to memory
8	BURSTEN	no effect on synchronous write
7	FWPRLVL	Set to protect memory from accidental writes
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 192. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-30	-	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	0 to get CLK = HCLK (not supported) 1 to get CLK = 2 × HCLK
19-16	BUSTURN	No effect
15-8	DATAST	No effect
7-4	ADDHLD	No effect
3-0	ADDSET	No effect

31.5.6 NOR/PSRAM controller registers

SRAM/NOR-Flash chip-select control registers 1..4 (FSMC_BCR1..4)

Address offset: 0xA000 0000 + 8 * (x - 1), x = 1...4

Reset value: 0x0000 30DX

This register contains the control information of each memory bank, used for SRAMs, ROMs and asynchronous or burst NOR Flash memories.

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

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

For Cellular RAM, the bit enables synchronous burst protocol during write operations. For Flash memory access in burst mode, this bit enables/disables the wait state insertion via the NWAIT signal. The enable bit for the synchronous burst protocol during read access is the BURSTEN bit in the FSMC_BCRx register.

0: Write operations are always performed in asynchronous mode

1: Write operations are performed in synchronous mode.

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

This bit enables the FSMC to use the wait signal even during an asynchronous protocol.

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

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

Bit 14 **EXTMOD:** Extended mode enable.

This bit enables the FSMC to program inside the FSMC_BWTR register, so it allows different timings for read and write.

0: values inside FSMC_BWTR register are not taken into account (default after reset)

1: values inside FSMC_BWTR register are taken into account

Bit 13 **WAITEN:** Wait enable bit.

For Flash memory access in burst mode, this bit enables/disables wait-state insertion via the NWAIT signal:

0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed Flash latency period)

1: NWAIT signal is enabled (its level is taken into account after the programmed Flash latency period to insert wait states if asserted) (default after reset)

Bit 12 **WRREN:** Write enable bit.

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

0: Write operations are disabled in the bank by the FSMC, an AHB error is reported,

1: Write operations are enabled for the bank by the FSMC (default after reset).

Bit 11 WAITCFG: Wait timing configuration.

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

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

Bit 10 WRAPMOD: Wrapped burst mode support.

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

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

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

Bit 9 WAITPOL: Wait signal polarity bit.

Defines the polarity of the wait signal from memory. Valid only when accessing the memory in burst mode:

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

Bit 8 BURSTEN: Burst enable bit.

Enables the burst access mode for the memory. Valid only with synchronous burst memories:

- 0: Burst access mode disabled (default after reset)
- 1: Burst access mode enable

Bit 7 Reserved, 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 databus width.

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

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

Bits 3:2 MTYP: Memory type.

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

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

Bit 1 MUXEN: Address/data multiplexing enable bit.

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

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

Bit 0 MBKEN: Memory bank enable bit.

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

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

SRAM/NOR-Flash chip-select timing registers 1..4 (FSMC_BTR1..4)

Address offset: 0xA000 0000 + 0x04 + 8 * (x - 1), x = 1..4

Reset value: 0xFFFF FFFF

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

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

Bits 29:28 ACCMOD: Access mode

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

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

Bits 27:24 DATLAT: Data latency for synchronous burst NOR Flash memory

For NOR Flash with synchronous burst mode enabled, defines the number of memory clock cycles (+2) to issue to the memory before getting the first data:

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

Note: This timing parameter is not expressed in HCLK periods, but in Flash clock (CLK) periods. In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care. In the case of CRAM, this field must be set to '0'.

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

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

- 0000: Reserved
 - 0001: CLK period = 2 × HCLK periods
 - 0010: CLK period = 3 × HCLK periods
 - 1111: CLK period = 16 × HCLK periods (default value after reset)
- In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care.

Bits 19:16 BUSTURN: Bus turnaround phase duration

These bits are written by software to insert the bus turnaround delay after a read access only from multiplexed NOR Flash memory to avoid bus contention if the controller needs to drive addresses on the databus for the next side-by-side transaction. BUSTURN can be set to the minimum if the slowest memory does not take more than 6 HCLK clock cycles to put the databus in Hi-Z state.

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

$(BUSTRUN + 1)\text{HCLK period} \geq t_{EHEL\min}$ and $(BUSTRUN + 2)\text{HCLK period} \geq t_{EHQZ\max}$ if EXTMOD = '0'

$(BUSTRUN + 2)\text{HCLK period} \geq \max(t_{EHEL\min}, t_{EHQZ\max})$ if EXTMOD = '1'.

0000: BUSTURN phase duration = 0 HCLK clock cycle added

...

1111: BUSTURN phase duration = $15 \times \text{HCLK clock cycles}$ (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 390](#) to [Figure 402](#)), used in SRAMs, ROMs and asynchronous NOR Flash accesses:

0000 0000: Reserved

0000 0001: DATAST phase duration = $1 \times \text{HCLK clock cycles}$

0000 0010: DATAST phase duration = $2 \times \text{HCLK clock cycles}$

...

1111 1111: DATAST phase duration = $255 \times \text{HCLK clock cycles}$ (default value after reset)

For each memory type and access mode data-phase duration, please refer to the respective figure ([Figure 390](#) to [Figure 402](#)).

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 399](#) to [Figure 402](#)), used in mode D and multiplexed accesses:

0000: Reserved

0001: ADDHLD phase duration = $1 \times \text{HCLK clock cycle}$

0010: ADDHLD phase duration = $2 \times \text{HCLK clock cycle}$

...

1111: ADDHLD phase duration = $15 \times \text{HCLK clock cycles}$ (default value after reset)

For each access mode address-hold phase duration, please refer to the respective figure ([Figure 399](#) to [Figure 402](#)).

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 390](#) to [Figure 402](#)), used in SRAMs, ROMs and asynchronous NOR Flash accesses:

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

...

1111: ADDSET phase duration = $1615 \times \text{HCLK clock cycles}$ (default value after reset)

For each access mode address setup phase duration, please refer to the respective figure (refer to [Figure 390](#) to [Figure 402](#)).

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

- Note:** PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to prolong the latency as needed.
- With PSRAMs (CRAMs) the field DATLAT must be set to 0, so that the FSMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write when the memory is ready.
- This method can be used also with the latest generation of synchronous Flash memories that issue the NWAIT signal, unlike older Flash memories (check the datasheet of the specific Flash memory being used).

SRAM/NOR-Flash write timing registers 1..4 (FSMC_BWTR1..4)

Address offset: 0xA000 0000 + 0x104 + 8 * (x - 1), x = 1...4

Reset value: 0xFFFF FFFF

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

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

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

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

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

Bits 27:24 **DATLAT:** Data latency (for synchronous burst NOR Flash).

For NOR Flash with Synchronous burst mode enabled, defines the number of memory clock cycles (+2) to issue to the memory before getting the first data:

0000: (0x0) Data latency of 2 CLK clock cycles for first burst access

...

1111: (0xF) Data latency of 17 CLK clock cycles for first burst access (default value after reset)

*Note: This timing parameter is not expressed in HCLK periods, but in Flash clock (**CLK**) periods. In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care. In case of CRAM, this field must be set to 0*

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

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

0000: Reserved

0001 CLK period = 2 × HCLK periods

0010 CLK period = 3 × HCLK periods

1111: CLK period = 16 × HCLK periods (default value after reset)

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

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

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

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

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 390](#) to [Figure 402](#)), used in SRAMs, ROMs and asynchronous NOR Flash accesses:

0000 0000: Reserved

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

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

...

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

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

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

These bits are written by software to define the duration of the *address hold* phase (refer to [Figure 399](#) to [Figure 402](#)), used in SRAMs, ROMs and asynchronous multiplexed NOR Flash accesses:

0000: Reserved

0001: ADDHLD phase duration = 1 × HCLK clock cycle

0010: ADDHLD phase duration = 2 × HCLK clock cycle

...

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

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

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

These bits are written by software to define the duration of the *address setup* phase in HCLK cycles (refer to [Figure 399](#) to [Figure 402](#)), used in SRAMs, ROMs and asynchronous NOR Flash accessed:

0000: ADDSET phase duration = 0 × HCLK clock cycle

...

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

Note: In synchronous NOR Flash accesses, this value is don't care.

31.6 NAND Flash/PC Card controller

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

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

The NAND/PC Card controller can control three external banks. Bank 2 and bank 3 support NAND Flash devices. Bank 4 supports PC Card devices.

Each bank is configured by means of dedicated registers ([Section 31.6.8](#)). The programmable memory parameters include access timings (shown in [Table 193](#)) and ECC configuration.

Table 193. Programmable NAND/PC Card access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Memory setup time	Number of clock cycles (HCLK) to set up the address before the command assertion	Read/Write	AHB clock cycle (HCLK)	1	256
Memory wait	Minimum duration (HCLK clock cycles) of the command assertion	Read/Write	AHB clock cycle (HCLK)	2	256
Memory hold	Number of clock cycles (HCLK) to hold the address (and the data in case of a write access) 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 databus is kept in high-Z state after the start of a write access	Write	AHB clock cycle (HCLK)	0	255

31.6.1 External memory interface signals

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

Caution: When using a PC Card or a CompactFlash in I/O mode, the NIOS16 input pin must remain at ground level during the whole operation, otherwise the FSMC may not operate properly. This means that the NIOS16 input pin must *not* be connected to the card, but directly to ground (only 16-bit accesses are allowed).

Note: *Prefix "N". specifies the associated signal as active low.*

8-bit NAND Flash

Table 194. 8-bit NAND Flash

FSMC 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 FSMC

There is no theoretical capacity limitation as the FSMC can manage as many address cycles as needed.

16-bit NAND Flash

Table 195. 16-bit NAND Flash

FSMC 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 FSMC

There is no theoretical capacity limitation as the FSMC can manage as many address cycles as needed.

Table 196. 16-bit PC Card

FSMC signal name	I/O	Function
A[10:0]	O	Address bus
NIOS16	I	Data transfer in I/O space. It must be shorted to GND (16-bit transfer only)
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 FSMC (memory signal name IORDY)
INTR	I	PC Card interrupt to the FSMC (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 31.3: AHB interface

31.6.2 NAND Flash / PC Card supported memories and transactions

Table 197 below shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND Flash / PC Card controller appear in gray.

Table 197. Supported memories and transactions

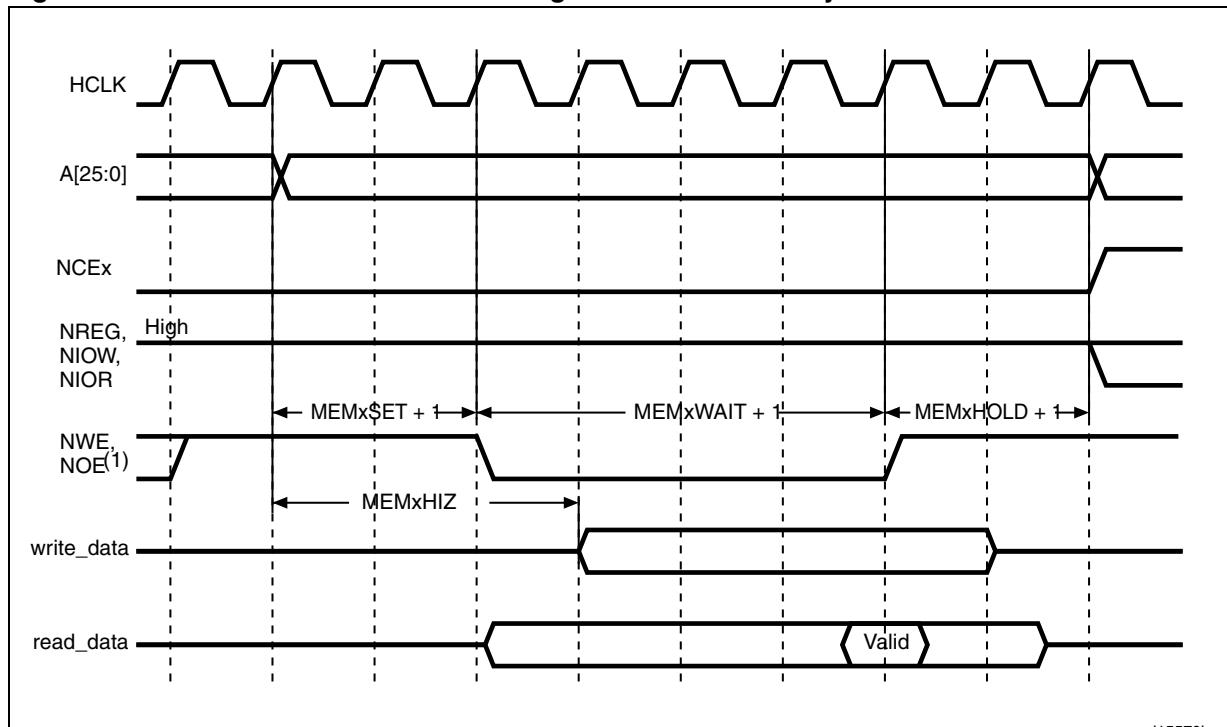
Device	Mode	R/W	AHB data size	Memory data size	Allowed/not allowed	Comments
NAND 8-bit	Asynchronous	R	8	8	Y	
	Asynchronous	W	8	8	Y	
	Asynchronous	R	16	8	Y	Split into 2 FSMC accesses
	Asynchronous	W	16	8	Y	Split into 2 FSMC accesses
	Asynchronous	R	32	8	Y	Split into 4 FSMC accesses
	Asynchronous	W	32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

31.6.3 Timing diagrams for NAND and PC Card

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

- Control register: FSMC_PCRx
- Interrupt status register: FSMC_SRx
- ECC register: FSMC_ECCRx
- Timing register for Common memory space: FSMC_PMEMx
- Timing register for Attribute memory space: FSMC_PATTx
- Timing register for I/O space: FSMC_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 databus in the case of a write. *Figure 408* 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 408. NAND/PC Card controller timing for common memory access

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

31.6.4 NAND Flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND Flash device are driven by some address signals of the FSMC 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 certain address in its memory space.

A typical page read operation from the NAND Flash device is as follows:

1. Program and enable the corresponding memory bank by configuring the `FSMC_PCRx` and `FSMC_PMEMx` (and for some devices, `FSMC_PATTx`, see [Section 31.6.5: NAND Flash pre-wait functionality on page 1266](#)) registers according to the characteristics of the NAND Flash (PWID bits for the databus width of the NAND Flash, PTYP = 1, PWAITEN = 1, PBKEN = 1, see section [Common memory space timing register 2.4 \(FSMC_PMEM2..4\) on page 1272](#) for timing configuration).
2. The CPU performs a byte write in the common memory space, with data byte equal to one Flash command byte (for example 0x00 for Samsung NAND Flash devices). The CLE input of the NAND Flash is active during the write strobe (low pulse on NWE), thus the written byte is interpreted as a command by the NAND Flash. Once the command is latched by the NAND Flash device, it does not need to be written for the following page read operations.
3. 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], then STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] for 64 Mb x 8 bit NAND Flash) 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

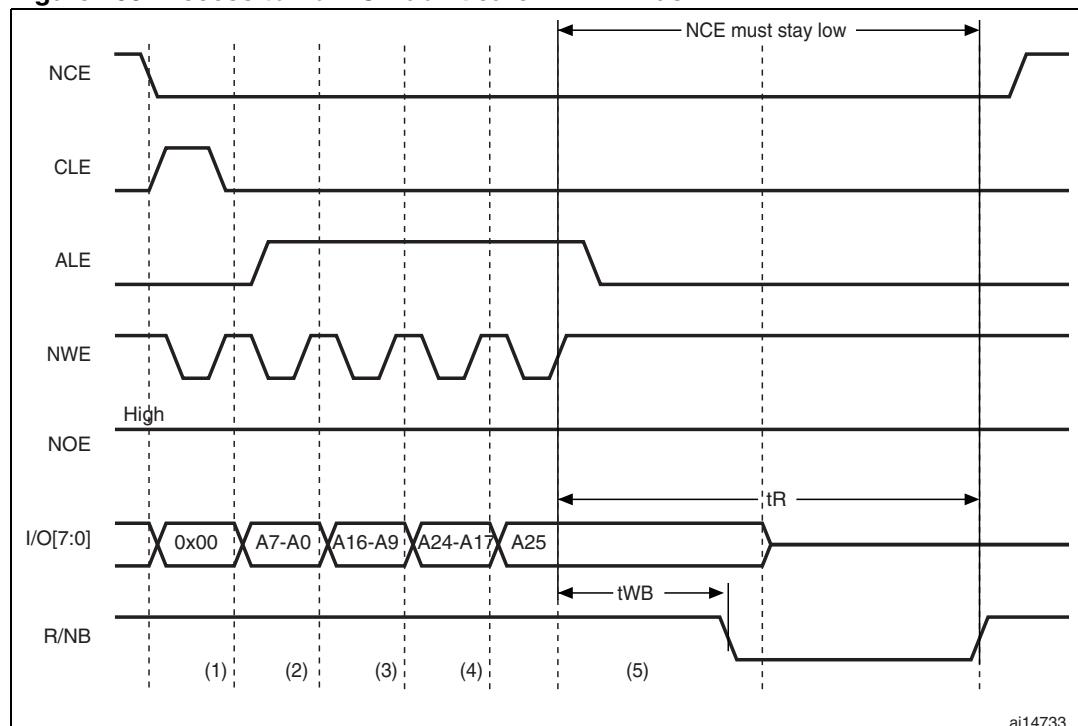
possible to use a different timing configuration of the FSMC, which can be used to implement the prewait functionality needed by some NAND Flash memories (see details in [Section 31.6.5: NAND Flash pre-wait functionality on page 1266](#)).

4. The controller waits for the NAND Flash to be ready (R/NB signal high) to become active, before starting a new access (to same or another memory bank). While waiting, the controller maintains the NCE signal active (low).
5. The CPU can then perform byte read operations in the common memory space to read the NAND Flash page (data field + Spare field) byte by byte.
6. The next NAND Flash page can be read without any CPU command or address write operation, 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

31.6.5 NAND Flash pre-wait functionality

Some NAND Flash devices require that, after writing the last part of the address, the controller wait for the R/NB signal to go low as shown in [Figure 409](#).

Figure 409. 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: FSMC performs a write access using `FSMC_PATT2` timing definition, where $\text{ATT HOLD} \geq 7$ (providing that $(7+1) \times \text{HCLK} = 112 \text{ ns} > t_{WB} \text{ max}$). 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 needed, it can be guaranteed by programming the MEMHOLD value to meet the t_{WB} timing, however any CPU read or write access to the NAND Flash then has the hold delay of (MEMHOLD + 1) HCLK cycles inserted from the rising edge of the NWE signal to the next access.

To overcome 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 leaving the MEMHOLD value at its minimum. Then, the CPU must 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.

31.6.6 Error correction code computation ECC (NAND Flash)

The FSMC PC-Card controller includes two error correction code computation hardware blocks, one per memory bank. They are used to reduce the host CPU workload when processing the error correction code by software in the system.

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

The error correction code (ECC) algorithm implemented in the FSMC 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 from or written to NAND Flash.

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

The functional operations are:

- When access to NAND Flash is made to bank 2 or bank 3, the data present on the D[15:0] bus is latched and used for ECC computation.
- When access to NAND Flash occurs at any other address, the ECC logic is idle, and does not perform any operation. Thus, write operations for defining commands or addresses to NAND Flash are not taken into account for ECC computation.

Once the desired number of bytes has been read from/written to the NAND Flash by the host CPU, the FSMC_ECCR2/3 registers must be read in order to retrieve the computed value. Once read, they should be cleared by resetting the ECCEN bit to zero. To compute a new data block, the ECCEN bit must be set to one in the FSMC_PCR2/3 registers.

31.6.7 PC Card/CompactFlash operations

Address spaces & memory accesses

The FSMC supports Compact Flash storage or PC Cards in Memory Mode and I/O Mode (True IDE mode is not supported).

The Compact Flash storage and PC Cards are made of 3 memory spaces:

- Common Memory Space
- Attribute Space
- I/O Memory Space

The nCE2 and nCE1 pins (FSMC_NCE4_2 and FSMC_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 low nIORD 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 during accesses to 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 will not lead to the low assertion of nCE2. 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 must be limited to AHB 16 bit accesses.

Table 198. 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
	YES										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	Invalid Attribute Space	Read or Write (odd address)	YES
0	1	0	0	1	X	X	X-X	x		Read or Write (odd address)	YES

Table 198. 16-bit PC-Card signals and access type (continued)

nCE2	nCE1	nREG	nOE/nWE	nIORD /nIOWR	A10	A9	A7-1	A0	Space	Access Type	Allowed/not Allowed
1	0	0	1	0	X	X	X-X	0	I/O space	Read Even Byte on D7-0	Not supported
1	0	0	1	0	X	X	X-X	1		Read Odd Byte on D7-0	Not supported
1	0	0	1	0	X	X	X-X	0		Write Even Byte on D7-0	Not supported
1	0	0	1	0	X	X	X-X	1		Write Odd Byte on D7-0	Not supported
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

The FSMC Bank 4 gives access to those 3 memory spaces as described in [Section 31.4.2: NAND/PC Card address mapping - Table 165: Memory mapping and timing registers](#)

Wait Feature

The CompactFlash Storage or PC Card may request the FSMC 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 FSMC_PCRx register. In order to detect the nWAIT assertion correctly, the MEMWAITx/ATTWAITx/IOWAITx bits must be programmed as follows:

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

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

After the de-assertion of nWAIT, the FSMC extends the WAIT phase for 4 HCLK clock cycles.

31.6.8 NAND Flash/PC Card controller registers

PC Card/NAND Flash control registers 2..4 (FSMC_PCR2..4)

Address offset: 0xA0000000 + 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.	ECEN		PWID			PTYP		PBKEN		PWAITEN		Reserved

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: $t_{ar} = (TAR + SET +) \times THCLK$ 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 $t_{clr} = (TCLR + SET +) \times THCLK$ 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..

Bits 6 **ECEN:** ECC computation logic enable bit

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

1: ECC logic is enabled.

Bits 5:4 **PWID:** Databus width.

Defines the external memory device width.

00: 8 bits (default after reset)

01: 16 bits (mandatory for PC Card)

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 nIORD/nIOWR is low.

Bit 0 Reserved, must be kept at reset value..

FIFO status and interrupt register 2..4 (FSMC_SR2..4)

Address offset: 0xA000 0000 + 0x44 + 0x20 * (x-1), x = 2..4

Reset value: 0x0000 0040

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

This is used to quickly write to the AHB and free it for transactions to peripherals other than the FSMC, while the FSMC is draining its FIFO into the memory. This register has one of its bits that indicates the status of the FIFO, for ECC purposes.

The ECC is calculated while the data are written to the memory, so in order to read the correct ECC the software must 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																									FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
r	rw	rw	rw	rw	rw	rw																									

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 (FSMC_PMEM2..4)

Address offset: Address: 0xA000 0000 + 0x48 + 0x20 * (x - 1), x = 2..4

Reset value: 0xFCFC FCFC

Each FSMC_PMEMx (x = 2..4) read/write register contains the timing information for PC Card or NAND Flash memory bank x, used for access to the common memory space of the 16-bit PC Card/CompactFlash, or to access 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		

Bits 31:24 **MEMHIZx:** Common memory x databus HiZ time

Defines the number of HCLK clock cycles during which the databus is kept in HiZ after the start of a PC Card/NAND Flash write access to common memory space on socket x. Only valid for write transaction:

0000 0000: (0x00) 0 HCLK cycle (for PC Card)

1111 1111: (0xFF) 255 HCLK cycles (for PC Card) - (default value after reset)

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

Defines the number of HCLK clock cycles to hold address (and data for write access) after the command deassertion (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 (default value after reset)

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 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: 2HCLK cycles (+ wait cycle introduced by deasserting NWAIT)

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

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

Defines the number of HCLK () 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) - (default value after reset)

Attribute memory space timing registers 2..4 (FSMC_PATT2..4)

Address offset: 0xA000 0000 + 0x4C + 0x20 * (x – 1), x = 2..4

Reset value: 0xFCFC FCFC

Each FSMC_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 31.6.5: NAND Flash pre-wait 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
ATTHIZx							ATTHOLDx							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		

Bits 31:24 **ATTHIZx:** Attribute memory x databus HiZ time

Defines the number of HCLK clock cycles during which the databus is kept in HiZ 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 (default value after reset)

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

Defines the number of HCLK clock cycles to hold address (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 (default value after reset)

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) (default value after reset)

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 (default value after reset)

I/O space timing register 4 (FSMC_PIO4)

Address offset: 0xA000 0000 + 0xB0

Reset value: 0xFCFCFCFC

The FSMC_PIO4 read/write registers contain the timing information used to gain access to 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
IOHZx							IOHOLDx							IOWAITx							IOSETx										
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 IOHZx: I/O x databus HiZ time

Defines the number of HCLK clock cycles during which the databus is kept in HiZ 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 (default value after reset)

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

Defines the number of HCLK clock cycles to hold address (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 (default value after reset)

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) (default value after reset)

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 (default value after reset)

ECC result registers 2/3 (FSMC_ECCR2/3)

Address offset: 0xA000 0000 + 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 FSMC 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 31.6.6: Error correction code computation ECC \(NAND Flash\)](#)), the data read from or written to the NAND Flash are processed automatically by ECC computation module. At the end of X bytes read (according to the ECCPS field in the FSMC_PCRx registers), the CPU must read the computed ECC value from the FSMC_ECCx registers, and then verify whether 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 if applicable. The FSMC_ECCR_x registers should be cleared after being read by setting the ECCEN bit to zero. For computing a new data block, the ECCEN bit must be set to one.

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 provides the value computed by the ECC computation logic. [Table 199](#) hereafter describes the contents of these bit fields.

Table 199. 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]

31.6.9 FSMC register map

The following table summarizes the FSMC registers.

Table 200. FSMC register map

Table 200. FSMC 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	
0xA000 00A8	FSMC_PMEM4						MEMHIZx						MEMHOLDx					MEMWAITx										MEMSETx						
0xA000 006C	FSMC_PATT2						ATTHIZx						ATTHOLDx					ATTWAITx										ATTSETx						
0xA000 008C	FSMC_PATT3						ATTHIZx						ATTHOLDx					ATTWAITx										ATTSETx						
0xA000 00AC	FSMC_PATT4						ATTHIZx						ATTHOLDx					ATTWAITx										ATTSETx						
0xA000 00B0	FSMC_PIO4						IOHIZx						IOHOLDx					IOWAITx										IOSETx						
0xA000 0074	FSMC_ECCR2																	ECCx																
0xA000 0094	FSMC_ECCR3																	ECCx																

Refer to [Table 1 on page 50](#) for the register boundary addresses.

32 Debug support (DBG)

32.1 Overview

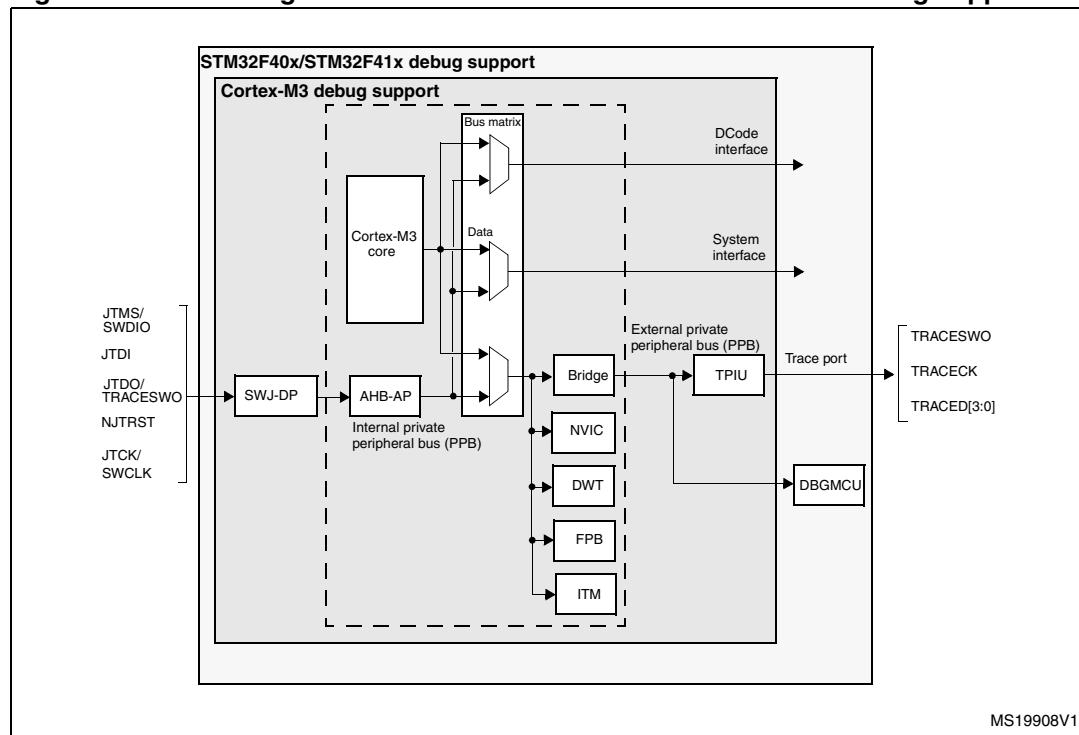
The STM32F40x and STM32F41x are built around a CortexTM-M4F core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state 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 STM32F40x and STM32F41x MCUs.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

Figure 410. Block diagram of STM32 MCU and CortexTM-M4F-level debug support



Note: The debug features embedded in the CortexTM-M4F core are a subset of the ARM CoreSight Design Kit.

The ARM Cortex™-M4F 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 on larger packages, where the corresponding pins are mapped)

It also includes debug features dedicated to the STM32F40x and STM32F41x:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

Note:

For further information on debug functionality supported by the ARM Cortex™-M4F core, refer to the Cortex™-M4F-r0p1 Technical Reference Manual and to the CoreSight Design Kit-r0p1 TRM (see [Section 32.2: Reference ARM documentation](#)).

32.2 Reference ARM documentation

- Cortex™-M4F r0p1 Technical Reference Manual (TRM)
(see Related documents on page 1)
- ARM Debug Interface V5
- ARM CoreSight Design Kit revision r0p1 Technical Reference Manual

32.3 SWJ debug port (serial wire and JTAG)

The STM32F40x and STM32F41x 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.

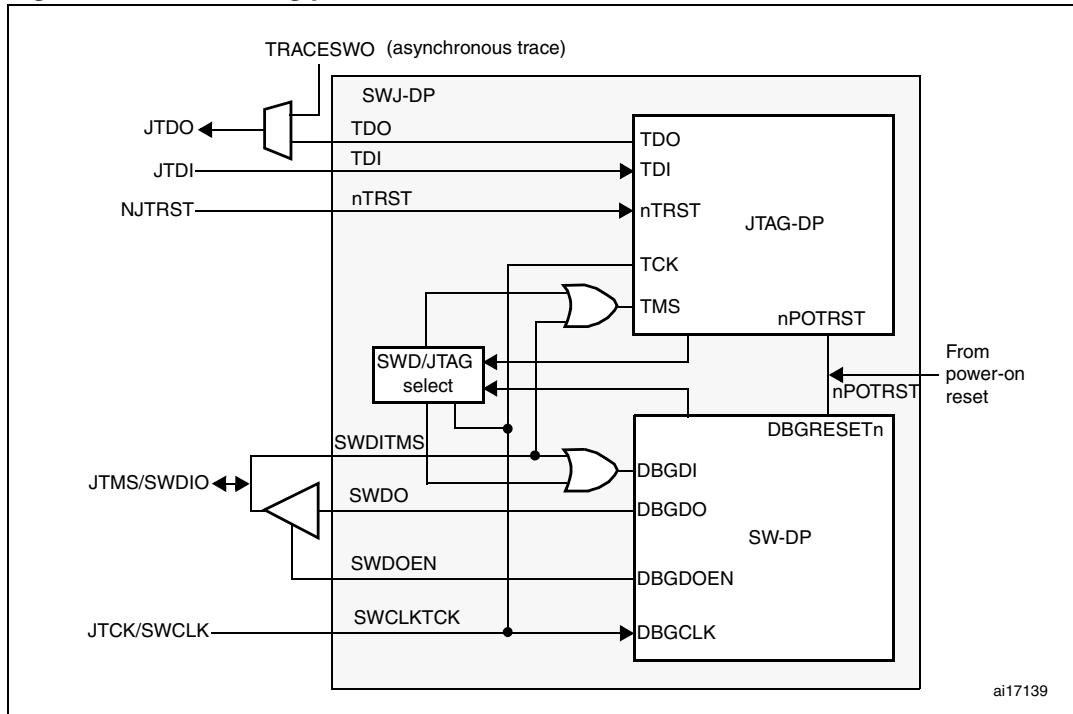
Figure 411. SWJ debug port

Figure 411 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.

32.3.1 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

32.4 Pinout and debug port pins

The STM32F40x and STM32F41x 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.

32.4.1 SWJ debug port pins

Five pins are used as outputs from the STM32F40x and STM32F41x for the SWJ-DP as *alternate functions* of general-purpose I/Os. These pins are available on all packages.

Table 201. 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

32.4.2 Flexible SWJ-DP pin assignment

After RESET (SYSRESETn or PORESETn), all five pins used for the SWJ-DP are assigned as dedicated pins immediately usable by the debugger host (note that the trace outputs are not assigned except if explicitly programmed by the debugger host).

However, the STM32F40x and STM32F41x MCUs offer the possibility of disabling some or all of the SWJ-DP ports and so, of releasing the associated pins for general-purpose IO (GPIO) usage. For more details on how to disable SWJ-DP port pins, please refer to [Section 6.3.2: I/O pin multiplexer and mapping](#).

Table 202. 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 GPIO_AFR 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.).

32.4.3 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.

32.4.4 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.

32.5 STM32F40x and STM32F41x JTAG TAP connection

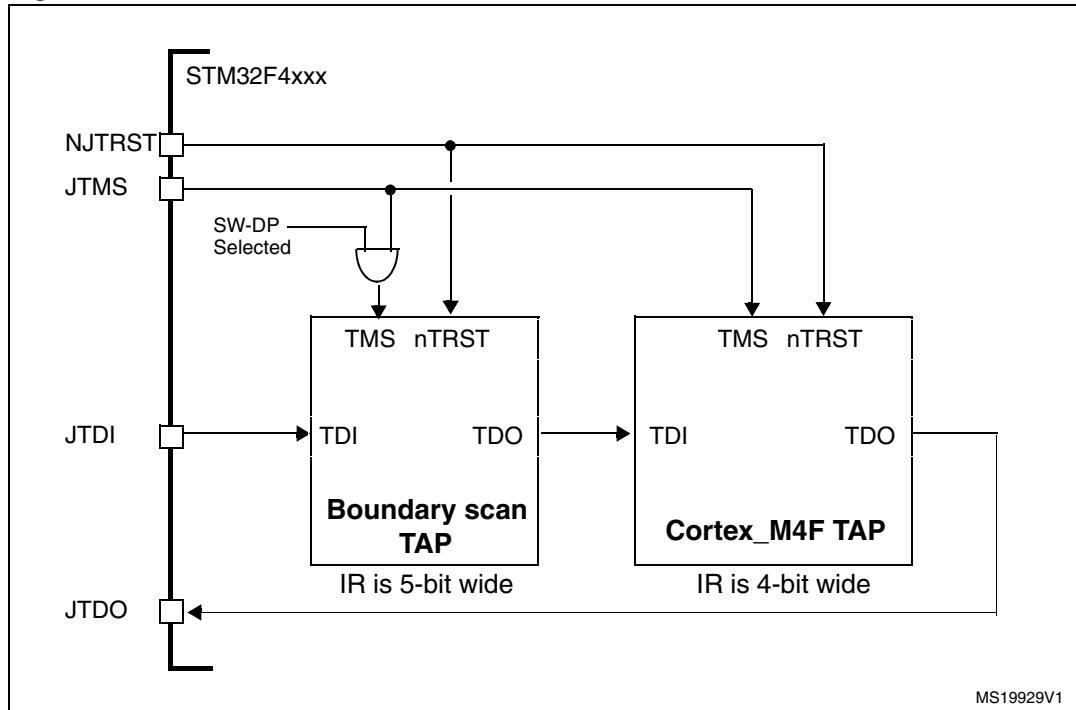
The STM32F40x and STM32F41x MCUs integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the CortexTM-M4F TAP (IR is 4-bit wide).

To access the TAP of the CortexTM-M4F 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 412. JTAG TAP connections



32.6 ID codes and locking mechanism

There are several ID codes inside the STM32F40x and STM32F41x 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.

32.6.1 MCU device ID code

The STM32F40x and STM32F41x 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 32.16 on page 1296](#)). 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
Reserved				DEV_ID											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID(15:0)** Revision identifier

This field indicates the revision of the device:

0x1000 = Revision A

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

Bits 11:0 **DEV_ID(11:0)**: Device identifier

The device ID is 0x413

32.6.2 Boundary scan TAP

JTAG ID code

The TAP of the STM32F40x and STM32F41x BSC (boundary scan) integrates a JTAG ID code equal to 0x06413041.

32.6.3 CortexTM-M4F TAP

The TAP of the ARM CortexTM-M4F 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 CortexTM-M4F r0p1, see [Section 32.2: Reference ARM documentation](#)).

Only the DEV_ID(11:0) should be used for identification by the debugger/programmer tools.

32.6.4 CortexTM-M4F JEDEC-106 ID code

The ARM CortexTM-M4F 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.

32.7 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 CortexTM-M4Fr0p1 Technical Reference Manual (*TRM*), for references, please see [Section 32.2: Reference ARM documentation](#)).

Table 203. JTAG debug port data registers

IR(3:0)	Data register	Details
1111	BYPASS [1 bit]	
1110	IDCODE [32 bits]	<i>ID CODE</i> 0x4BA00477 (ARM Cortex TM -M4F r0p1 ID Code)

Table 203. JTAG debug port data registers (continued)

IR(3:0)	Data register	Details
1010	DPACC [35 bits]	<p><i>Debug port access register</i></p> <p>This initiates a debug port and allows access to a debug 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 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: <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>Refer to Table 204 for a description of the A(3:2) bits</p>
1011	APACC [35 bits]	<p><i>Access port access register</i></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><i>Abort register</i></p> <ul style="list-style-type: none"> – Bits 31:1 = Reserved – Bit 0 = DAPABORT: write 1 to generate a DAP abort.

Table 204. 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)

Table 204. 32-bit debug port registers addressed through the shifted value A[3:2]

Address	A(3:2) value	Description
0x8	10	DP SELECT register: Used to select the current access port and the active 4-words register window. – Bits 31:24: APSEL: select the current AP – Bits 23:8: reserved – Bits 7:4: APBANKSEL: select the active 4-words register window on the current AP – Bits 3:0: reserved
0xC	11	DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)

32.8 SW debug port

32.8.1 SW protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO bidirectional management, the line must be pulled-up on the board (100 KΩ recommended by ARM).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

32.8.2 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 205. 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

Table 205. Packet request (8-bits) (continued)

Bit	Name	Description
4:3	A(3:2)	Address field of the DP or AP registers (refer to Table 204)
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™-M4Fr0p1 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 206. 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 207. 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.

32.8.3 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 **0x2BA01477** (corresponding to Cortex™-M4F 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™-M4F r0p1 TRM* and the *CoreSight Design Kit r0p1 TRM*.

32.8.4 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.

32.8.5 SW-DP registers

Access to these registers are initiated when APnDP=0

Table 208. 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: – 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.

Table 208. SW-DP registers (continued)

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
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

32.8.6 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

32.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP

Features:

- System access is independent of the processor status.
- Either SW-DP or JTAG-DP accesses AHB-AP.
- The AHB-AP is an AHB master into the Bus Matrix. Consequently, it can access all the data buses (Dcode Bus, System Bus, internal and external PPB bus) but the ICode bus.
- Bitband transactions are supported.
- AHB-AP transactions bypass the FPB.

The address of the 32-bits AHP-AP registers are 6-bits wide (up to 64 words or 256 bytes) and consists of:

- c) Bits [7:4] = the bits [7:4] APBANKSEL of the DP SELECT register
- d) 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™-M4F includes 9 x 32-bits registers:

Table 209. Cortex™-M4F 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	
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™-M4F r0p1 TRM* for further details.

32.10 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 210. Core debug registers

Register	Description
DHCSR	<i>The 32-bit Debug Halting Control and Status Register</i> This provides status information about the state of the processor enable core debug halt and step the processor
DCRSR	<i>The 17-bit Debug Core Register Selector Register</i> : This selects the processor register to transfer data to or from.
DCRDR	<i>The 32-bit Debug Core Register Data Register</i> : This holds data for reading and writing registers to and from the processor selected by the DCRSR (Selector) register.
DEMCR	<i>The 32-bit Debug Exception and Monitor Control Register</i> : This provides Vector Catching and Debug Monitor Control. 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™-M4F 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.

32.11 Capability of the debugger host to connect under system reset

The STM32F40x and STM32F41x 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™-M4F 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.*

32.12 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.

32.13 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

32.14 ITM (instrumentation trace macrocell)

32.14.1 General description

The ITM is an application-driven trace source that supports *printf* style debugging to trace *Operating System* (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated as:

- **Software trace.** Software can write directly to the ITM stimulus registers to emit packets.
- **Hardware trace.** The DWT generates these packets, and the ITM emits them.
- **Time stamping.** Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex™-M4F clock or the bit clock rate of the *Serial Wire Viewer* (SWV) output clocks the counter.

The packets emitted by the ITM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to TPIU) and then output the complete packets sequence to the debugger host.

The bit TRCEN of the Debug Exception and Monitor Control Register must be enabled before you program or use the ITM.

32.14.2 Time stamp packets, synchronization and overflow packets

Time stamp packets encode time stamp information, generic control and synchronization. It uses a 21-bit timestamp counter (with possible prescalers) which is reset at each time stamp packet emission. This counter can be either clocked by the CPU clock or the SWV clock.

A synchronization packet consists of 6 bytes equal to 0x80_00_00_00_00_00 which is emitted to the TPIU as 00 00 00 00 00 80 (LSB emitted first).

A synchronization packet is a timestamp packet control. It is emitted at each DWT trigger.

For this, the DWT must be configured to trigger the ITM: the bit CYCCNTENA (bit0) of the DWT Control Register must be set. In addition, the bit2 (SYNCENA) of the ITM Trace Control Register must be set.

Note: *If the SYNCENA bit is not set, the DWT generates Synchronization triggers to the TPIU which will send only TPIU synchronization packets and not ITM synchronization packets.*

An overflow packet consists is a special timestamp packets which indicates that data has been written but the FIFO was full.

Table 211. Main ITM registers

Address	Register	Details
@E0000FB0	ITM lock access	Write 0xC5ACCE55 to unlock Write Access to the other ITM registers
@E0000E80	ITM trace control	Bits 31-24 = Always 0
		Bits 23 = Busy
		Bits 22-16 = 7-bits ATB ID which identifies the source of the trace data.
		Bits 15-10 = Always 0
		Bits 9:8 = TSPrescale = Time Stamp Prescaler
		Bits 7-5 = Reserved
		Bit 4 = SWOENA = Enable SWV behavior (to clock the timestamp counter by the SWV clock).
		Bit 3 = DWTENA: Enable the DWT Stimulus
		Bit 2 = SYNCENA: this bit must be to 1 to enable the DWT to generate synchronization triggers so that the TPIU can then emit the synchronization packets.
		Bit 1 = TSENA (Timestamp Enable)
		Bit 0 = ITMENA: Global Enable Bit of the ITM
@E0000E40	ITM trace privilege	Bit 3: mask to enable tracing ports31:24
		Bit 2: mask to enable tracing ports23:16
		Bit 1: mask to enable tracing ports15:8
		Bit 0: mask to enable tracing ports7:0
@E0000E00	ITM trace enable	Each bit enables the corresponding Stimulus port to generate trace.
@E0000000-E000007C	Stimulus port registers 0-31	Write the 32-bits data on the selected Stimulus Port (32 available) to be traced out.

Example of configuration

To output a simple value to the TPIU:

- Configure the TPIU and assign TRACE I/Os by configuring the DBGMCU_CR (refer to [Section 32.17.2: TRACE pin assignment](#) and [Section 32.16.3: Debug MCU configuration register](#))
- Write 0xC5ACCE55 to the ITM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00010005 to the ITM Trace Control Register to enable the ITM with Sync enabled and an ATB ID different from 0x00
- Write 0x1 to the ITM Trace Enable Register to enable the Stimulus Port 0
- Write 0x1 to the ITM Trace Privilege Register to unmask stimulus ports 7:0
- Write the value to output in the Stimulus Port Register 0: this can be done by software (using a printf function)

32.15 ETM (Embedded trace macrocell)

32.15.1 General description

The ETM enables the reconstruction of program execution. Data are traced using the Data Watchpoint and Trace (DWT) component or the Instruction Trace Macrocell (ITM) whereas instructions are traced using the Embedded Trace Macrocell (ETM).

The ETM transmits information as packets and is triggered by embedded resources. These resources must be programmed independently and the trigger source is selected using the Trigger Event Register (0xE0041008). An event could be a simple event (address match from an address comparator) or a logic equation between 2 events. The trigger source is one of the four comparators of the DWT module. The following events can be monitored:

- Clock cycle matching
- Data address matching

For more informations on the trigger resources refer to [Section 32.13: DWT \(data watchpoint trigger\)](#).

The packets transmitted by the ETM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to [Section 32.17: TPIU \(trace port interface unit\)](#)) and then outputs the complete packet sequence to the debugger host.

32.15.2 Signal protocol, packet types

This part is described in the chapter 7 ETMv3 Signal Protocol of the ARM IHI 0014N document.

32.15.3 Main ETM registers

For more information on registers refer to the chapter 3 of the ARM IHI 0014N specification.

Table 212. Main ETM registers

Address	Register	Details
0xE0041FB0	ETM Lock Access	Write 0xC5ACCE55 to unlock the write access to the other ETM registers.
0xE0041000	ETM Control	This register controls the general operation of the ETM, for instance how tracing is enabled.
0xE0041010	ETM Status	This register provides information about the current status of the trace and trigger logic.
0xE0041008	ETM Trigger Event	This register defines the event that will control trigger.
0xE004101C	ETM Trace Enable Control	This register defines which comparator is selected.
0xE0041020	ETM Trace Enable Event	This register defines the trace enabling event.
0xE0041024	ETM Trace Start/Stop	This register defines the traces used by the trigger source to start and stop the trace, respectively.

32.15.4 Configuration example

To output a simple value to the TPIU:

- Configure the TPIU and enable the I/O_TRACEN to assign TRACE I/Os in the STM32F40x and STM32F41x debug configuration register.
- Write 0xC5ACCE55 to the ETM Lock Access Register to unlock the write access to the ETM registers
- Write 0x00001D1E to the control register (configure the trace)
- Write 0000406F to the Trigger Event register (define the trigger event)
- Write 0000006F to the Trace Enable Event register (define an event to start/stop)
- Write 00000001 to the Trace Start/stop register (enable the trace)
- Write 0000191E to the ETM Control Register (end of configuration)

32.16 MCU debug component (DBGMCU)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog, I2C and bxCAN during a breakpoint
- Control of the trace pins assignment

32.16.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode, DBG_SLEEP bit of DBGMCU_CR register must be previously set by the debugger. This will feed HCLK with the same clock that is provided to FCLK (system clock previously configured by the software).
- In Stop mode, the bit DBG_STOP must be previously set by the debugger. This will enable the internal RC oscillator clock to feed FCLK and HCLK in STOP mode.

32.16.2 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.

32.16.3 Debug MCU configuration register

This register allows the configuration of the MCU under DEBUG. This concerns:

- Low-power mode support
- Timer and watchdog counter support
- bxCAN communication support
- Trace pin assignment

This DBGMCU_CR is mapped on the External PPB bus at address 0xE0042004

It is asynchronously reset by the PORRESET (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support these features, it is still possible for the user software to write to these registers.

DBGMCU_CR

Address: 0xE004 2004

Only 32-bit access supported

POR Reset: 0x0000 0000 (not reset by system reset)

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

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

Bits 7:5 **TRACE_MODE[1:0]** and **TRACE_IOEN**: Trace pin assignment control

- With *TRACE_IOEN*=0:
TRACE_MODE=xx: TRACE pins not assigned (default state)
- With *TRACE_IOEN*=1:
 - TRACE_MODE=00: TRACE pin assignment for Asynchronous Mode
 - TRACE_MODE=01: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 1
 - TRACE_MODE=10: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 2
 - TRACE_MODE=11: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 4

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Debug Standby mode

0: (FCLK=Off, HCLK=Off) The whole digital part is unpowered.

From software point of view, exiting from Standby is identical than fetching reset vector (except a few status bit indicated that the MCU is resuming from Standby)

1: (FCLK=On, HCLK=On) In this case, the digital part is not unpowered and FCLK and HCLK are provided by the internal RC oscillator which remains active. In addition, the MCU generate a system reset during Standby mode so that exiting from Standby is identical than fetching from reset

Bit 1 **DBG_STOP**: Debug Stop mode

0: (FCLK=Off, HCLK=Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI)). Consequently, the software must reprogram the clock controller to enable the PLL, the Xtal, etc.

1: (FCLK=On, HCLK=On) In this case, when entering STOP mode, FCLK and HCLK are provided by the internal RC oscillator which remains active in STOP mode. When exiting STOP mode, the software must reprogram the clock controller to enable the PLL, the Xtal, etc. (in the same way it would do in case of *DBG_STOP*=0)

Bit 0 **DBG_SLEEP**: Debug Sleep mode

0: (FCLK=On, HCLK=Off) In Sleep mode, FCLK is clocked by the system clock as previously configured by the software while HCLK is disabled.

In Sleep mode, the clock controller configuration is not reset and remains in the previously programmed state. Consequently, when exiting from Sleep mode, the software does not need to reconfigure the clock controller.

1: (FCLK=On, HCLK=On) In this case, when entering Sleep mode, HCLK is fed by the same clock that is provided to FCLK (system clock as previously configured by the software).

32.16.4 Debug MCU APB1 freeze register (DBGMCU_APB1_FZ)

The DBGMCU_APB1_FZ register is used to configure the MCU under Debug. It concerns APB1 peripherals. It is mapped on the external PPB bus at address 0xE004 2008.

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-bits 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	
Reserved					DBG_CAN2_STOP	DBG_CAN1_STOP	Reserved	DBG_I2C3_SMBUS_TIMEOUT	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT						
15	14	13	12	11	10	9		rw	rw	rw						
Reserved		DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Reserved	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	rw	
		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

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

Bit 26 **DBG_CAN2_STOP:** Debug CAN2 stopped when Core is halted

- 0: Same behavior as in normal mode
- 1: The CAN2 receive registers are frozen

Bit 25 **DBG_CAN1_STOP:** Debug CAN2 stopped when Core is halted

- 0: Same behavior as in normal mode
- 1: The CAN2 receive registers are frozen

Bit 24 Reserved, must be kept at reset value.

Bit 23 **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

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

- Bit 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:** RTC stopped when Core is halted
0: The RTC counter clock continues even if the core is halted
1: The RTC counter clock is stopped when the core is halted
- Bit 9 Reserved, must be kept at reset value.
- Bits 8:0 **DBG_TIMx_STOP:** TIMx counter stopped when core is halted (x=2..7, 12..14)
0: The clock of the involved Timer Counter is fed even if the core is halted
1: The clock of the involved Timer counter is stopped when the core is halted

32.16.5 Debug MCU APB2 Freeze register (DBGMCU_APB2_FZ)

The DBGMCU_APB2_FZ register is used to configure the MCU under Debug. It concerns APB2 peripherals.

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	
Reserved														DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP
														rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved														DBG_TIM8_STOP	DBG_TIM1_STOP	
														rw	rw	

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

Bits 18:16 **DBG_TIMx_STOP:** TIMx counter stopped when core is halted (x=9..11)

- 0: The clock of the involved Timer Counter is fed even if the core is halted
- 1: The clock of the involved Timer counter is stopped when the core is halted

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

Bit 1 **DBG_TIM8_STOP:** TIM8 counter stopped when core is halted

- 0: The clock of the involved Timer Counter is fed even if the core is halted
- 1: The clock of the involved Timer counter is stopped when the core is halted

Bit 0 **DBG_TIM1_STOP**: TIM1 counter stopped when core is halted

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

32.17 TPIU (trace port interface unit)

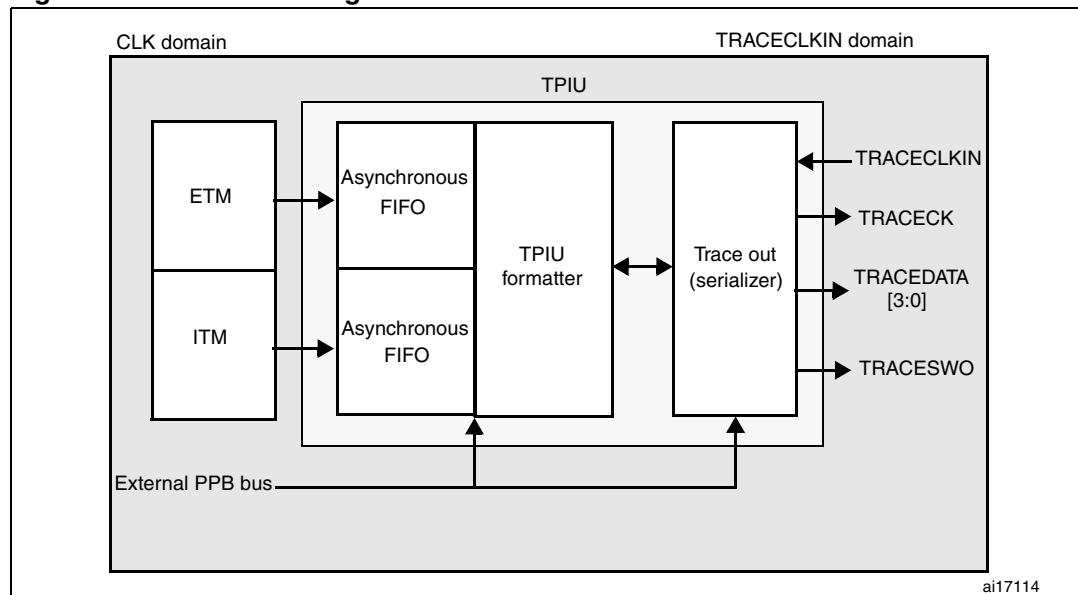
32.17.1 Introduction

The TPIU acts as a bridge between the on-chip trace data from the ITM and the ETM.

The output data stream encapsulates the trace source ID, that is then captured by a *trace port analyzer* (TPA).

The core embeds a simple TPIU, especially designed for low-cost debug (consisting of a special version of the CoreSight TPIU).

Figure 413. TPIU block diagram



32.17.2 TRACE pin assignment

- Asynchronous mode

The asynchronous mode requires 1 extra pin and is available on all packages. It is only available if using Serial Wire mode (not in JTAG mode).

Table 213. Asynchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32F40x and STM32F41x 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 214. Synchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32F40x and STM32F41x pin assignment
	Type	Description	
TRACECK	O	TRACE Clock	PE2
TRACED[3:0]	O	TRACE Sync Data Outputs Can be 1, 2 or 4.	PE[6:3]

TPUI TRACE pin assignment

By default, these pins are NOT assigned. They can be assigned by setting the TRACE_IOEN and TRACE_MODE bits in the ***MCU Debug component configuration register***. This configuration has to be done by the debugger host.

In addition, the number of pins to assign depends on the trace configuration (asynchronous or synchronous).

- **Asynchronous mode:** 1 extra pin is needed
- **Synchronous mode:** from 2 to 5 extra pins are needed depending on the size of the data trace port register (1, 2 or 4):
 - TRACECK
 - TRACED(0) if port size is configured to 1, 2 or 4
 - TRACED(1) if port size is configured to 2 or 4
 - TRACED(2) if port size is configured to 4
 - TRACED(3) if port size is configured to 4

To assign the TRACE pin, the debugger host must program the bits TRACE_IOEN and TRACE_MODE[1:0] of the Debug MCU configuration Register (DBGMCU_CR). By default the TRACE pins are not assigned.

This register is mapped on the external PPB and is reset by the PORESET (and not by the SYSTEM reset). It can be written by the debugger under SYSTEM reset.

Table 215. 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]				
	10	Synchronous Trace 2 bit		TRACECK	TRACED[0]	TRACED[1]			
	11	Synchronous Trace 4 bit		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]	

1. When Serial Wire mode is used, it is released. But when JTAG is used, it is assigned to JTDO.

Note: By default, the TRACECLKIN input clock of the TPIU is tied to GND. It is assigned to HCLK two clock cycles after the bit TRACE_IOEN has been set.

The debugger must then program the Trace Mode by writing the PROTOCOL[1:0] bits in the SPP_R (Selected Pin Protocol) register of the TPIU.

- PROTOCOL=00: Trace Port Mode (synchronous)
- PROTOCOL=01 or 10: Serial Wire (Manchester or NRZ) Mode (asynchronous mode). Default state is 01

It then also configures the TRACE port size by writing the bits [3:0] in the CPSPS_R (Current Sync Port Size Register) of the TPIU:

- 0x1 for 1 pin (default state)
- 0x2 for 2 pins
- 0x8 for 4 pins

32.17.3 TPUI formatter

The formatter protocol outputs data in 16-byte frames:

- seven bytes of data
- eight bytes of mixed-use bytes consisting of:
 - 1 bit (LSB) to indicate it is a DATA byte ('0) or an ID byte ('1).
 - 7 bits (MSB) which can be data or change of source ID trace.
- one byte of auxiliary bits where each bit corresponds to one of the eight mixed-use bytes:
 - if the corresponding byte was a data, this bit gives bit0 of the data.
 - if the corresponding byte was an ID change, this bit indicates when that ID change takes effect.

Note: Refer to the ARM CoreSight Architecture Specification v1.0 (ARM IHI 0029B) for further information

32.17.4 TPUI frame synchronization packets

The TPUI can generate two types of synchronization packets:

- The Frame Synchronization packet (or Full Word Synchronization packet)

It consists of the word: 0x7F_FF_FF_FF (LSB emitted first). This sequence can not occur at any other time provided that the ID source code 0x7F has not been used.

It is output periodically ***between*** frames.

In continuous mode, the TPA must discard all these frames once a synchronization frame has been found.

- The Half-Word Synchronization packet

It consists of the half word: 0x7F_FF (LSB emitted first).

It is output periodically ***between or within*** frames.

These packets are only generated in continuous mode and enable the TPA to detect that the TRACE port is in IDLE mode (no TRACE to be captured). When detected by the TPA, it must be discarded.

32.17.5 Transmission of the synchronization frame packet

There is no Synchronization Counter register implemented in the TPIU of the core.

Consequently, the synchronization trigger can only be generated by the **DWT**. Refer to the registers DWT Control Register (bits SYNCTAP[11:10]) and the DWT Current PC Sampler Cycle Count Register.

The 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).

32.17.6 Synchronous mode

The trace data output size can be configured to 4, 2 or 1 pin: TRACED(3:0)

The output clock is output to the debugger (TRACECK)

Here, TRACECLKIN is driven internally and is connected to HCLK only when TRACE is used.

Note:

In this synchronous mode, it is not required to provide a stable clock frequency.

The TRACE I/Os (including TRACECK) are driven by the rising edge of TRACELKIN (equal to HCLK). Consequently, the output frequency of TRACECK is equal to HCLK/2.

32.17.7 Asynchronous mode

This is a low cost alternative to output the trace using only 1 pin: this is the asynchronous output pin TRACESWO. Obviously there is a limited bandwidth.

TRACESWO is multiplexed with JTDO when using the SW-DP pin. This way, this functionality is available in all STM32F40x and STM32F41x 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%.

32.17.8 TRACECLKIN connection inside the STM32F40x and STM32F41x

In the STM32F40x and STM32F41x, 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 STM32F40x and STM32F41x 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.

32.17.9 TPIU registers

The TPIU APB registers can be read and written only if the bit TRCENA of the Debug Exception and Monitor Control Register (DEMCR) is set. Otherwise, the registers are read as zero (the output of this bit enables the PCLK of the TPIU).

Table 216. Important TPIU registers

Address	Register	Description
0xE0040004	Current port size	<p>Allows the trace port size to be selected:</p> <p>Bit 0: Port size = 1 Bit 1: Port size = 2 Bit 2: Port size = 3, not supported Bit 3: Port Size = 4</p> <p>Only 1 bit must be set. By default, the port size is one bit. (0x00000001)</p>
0xE00400F0	Selected pin protocol	<p>Allows the Trace Port Protocol to be selected:</p> <p>Bit1:0=</p> <p>00: Sync Trace Port Mode 01: Serial Wire Output - manchester (default value) 10: Serial Wire Output - NRZ 11: reserved</p>
0xE0040304	Formatter and flush control	<p>Bit 31-9 = always '0 Bit 8 = TrigIn = always '1 to indicate that triggers are indicated Bit 7-4 = always 0 Bit 3-2 = always 0 Bit 1 = EnFCont. In Sync Trace mode (Select_Pin_Protocol register bit1:0=00), this bit is forced to '1: the formatter is automatically enabled in continuous mode. In asynchronous mode (Select_Pin_Protocol register bit1:0 <> 00), this bit can be written to activate or not the formatter. Bit 0 = always 0</p> <p>The resulting default value is 0x102</p> <p>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).</p>
0xE0040300	Formatter and flush status	Not used in Cortex™-M4F, always read as 0x00000008

32.17.10 Example of configuration

- Set the bit TRCENA in the Debug Exception and Monitor Control Register (DEMCR)
- Write the TPIU Current Port Size Register to the desired value (default is 0x1 for a 1-bit port size)
- Write TPIU Formatter and Flush Control Register to 0x102 (default value)
- Write the TPIU Select Pin Protocol to select the sync or async mode. Example: 0x2 for async NRZ mode (UART like)
- Write the DBGMCU control register to 0x20 (bit IO_TRACEN) to assign TRACE I/Os for async mode. A TPIU Sync packet is emitted at this time (FF_FF_FF_7F)
- Configure the ITM and write the ITM Stimulus register to output a value

32.18 DBG register map

The following table summarizes the Debug registers

Table 217. DBG register map and reset values

Addr.	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE004 2000	DBGMCU_IDC_ODE	REV_ID												Reserved	DEV_ID																		
		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
0xE004 2004	DBGMCU_CR	Reserved												Reserved	Reserved												TRAC_E_MODE [1:0]	FIO_Z[2:0]	Reserved	DBG_STANDBY	DBG_STOP	DBG_SLEEP	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xE004 2008	DBGMCU_APB1_FZ	Reserved												Reserved	Reserved												DBG_IWDG_STOP	DBG_WWDG_STOP	Reserved	DBG_RTC_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xE004 200C	DBGMCU_APB2_FZ	Reserved												Reserved	Reserved												DBG_TIM8_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_STOP	DBG_SLEEP	
		0	0	0	0	0	0	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. The reset value is product dependent. For more information, refer to [Section 32.6.1: MCU device ID code](#).