

## Introduction

This document is addressed to application developers. It provides complete information on how to use the STM32WB55xx and STM32WB35xx microcontrollers.

These multiprotocol wireless and ultra-low-power devices embed a powerful ultra-low-power radio compliant with the Bluetooth<sup>®</sup> Low Energy SIG specification 5.3 and with IEEE 802.15.4-2011. They contain a dedicated Arm<sup>®</sup> Cortex<sup>®</sup>-M0+ for performing the real-time low layer operation.

The STM32WB55xx and STM32WB35xx microcontrollers feature different memory sizes, packages and peripherals.

## Related documents

Available from STMicroelectronics web site [www.st.com](http://www.st.com):

- STM32WB55xx and STM32WB35xx datasheets
- STM32WB55xx and STM32WB35xx errata sheets

For information on the Arm<sup>®</sup> Cortex<sup>®</sup>-M4 and Cortex<sup>®</sup>-M0+ cores, refer, respectively, to the corresponding Technical Reference Manuals, available from the [www.arm.com](http://www.arm.com) website.

For information on 802.15.4 refer to the IEEE website ([www.ieee.org](http://www.ieee.org)).

For information on Bluetooth<sup>®</sup> refer to [www.bluetooth.com](http://www.bluetooth.com).

# Contents

<b>1</b>	<b>Documentation conventions</b>	<b>61</b>
1.1	General information	61
1.2	List of abbreviations for registers	61
1.3	Glossary	62
1.4	Availability of peripherals	62
<b>2</b>	<b>System and memory overview</b>	<b>63</b>
2.1	System architecture	63
2.1.1	S0: CPU1 (CPU1 Cortex®-M4) I-bus	64
2.1.2	S1: CPU1 (CPU1 Cortex®-M4) D-bus	64
2.1.3	S2: CPU1 (CPU1 Cortex®-M4) S-bus	64
2.1.4	S3: CPU2 (Cortex®-M0+) S-bus	65
2.1.5	S4, S5: DMA-bus	65
2.1.6	S6: Radio system-bus	65
2.1.7	BusMatrix	65
2.2	Memory organization	66
2.2.1	Introduction	66
2.2.2	Memory map and register boundary addresses	67
2.2.3	Bit banding	72
2.3	Boot configuration	73
2.4	CPU2 boot	74
2.5	CPU2 SRAM fetch disable	74
<b>3</b>	<b>Embedded flash memory (FLASH)</b>	<b>75</b>
3.1	Introduction	75
3.2	FLASH main features	75
3.3	FLASH functional description	75
3.3.1	Flash memory organization	75
3.3.2	Empty check	76
3.3.3	Error code correction (ECC)	77
3.3.4	Read access latency	77
3.3.5	Adaptive real-time memory accelerator (ART Accelerator)	79
3.3.6	Flash memory program and erase operations	82

3.3.7	Flash main memory erase sequences . . . . .	83
3.3.8	Flash main memory programming sequences . . . . .	84
3.4	FLASH option bytes . . . . .	90
3.4.1	Option bytes description . . . . .	90
3.4.2	Option bytes programming . . . . .	97
3.5	FLASH UID64 . . . . .	100
3.6	Flash memory protection . . . . .	101
3.6.1	Read protection (RDP) . . . . .	101
3.6.2	Proprietary code readout protection (PCROP) . . . . .	105
3.6.3	Write protection (WRP) . . . . .	106
3.6.4	CPU2 security (ESE) . . . . .	107
3.7	FLASH program/erase suspension . . . . .	108
3.8	FLASH interrupts . . . . .	109
3.9	Register access protection . . . . .	109
3.10	FLASH registers . . . . .	110
3.10.1	Flash memory access control register (FLASH_ACR) . . . . .	110
3.10.2	Flash memory key register (FLASH_KEYR) . . . . .	111
3.10.3	Flash memory option key register (FLASH_OPTKEYR) . . . . .	111
3.10.4	Flash memory status register (FLASH_SR) . . . . .	112
3.10.5	Flash memory control register (FLASH_CR) . . . . .	113
3.10.6	Flash memory ECC register (FLASH_ECCR) . . . . .	115
3.10.7	Flash memory option register (FLASH_OPTR) . . . . .	116
3.10.8	Flash memory PCROP zone A start address register (FLASH_PCROP1ASR) . . . . .	118
3.10.9	Flash memory PCROP zone A end address register (FLASH_PCROP1AER) . . . . .	119
3.10.10	Flash memory WRP area A address register (FLASH_WRP1AR) . . . . .	119
3.10.11	Flash memory WRP area B address register (FLASH_WRP1BR) . . . . .	120
3.10.12	Flash memory PCROP zone B start address register (FLASH_PCROP1BSR) . . . . .	120
3.10.13	Flash memory PCROP zone B end address register (FLASH_PCROP1BER) . . . . .	121
3.10.14	Flash memory IPCC mailbox data buffer address register (FLASH_IPCCBR) . . . . .	121
3.10.15	Flash memory CPU2 access control register (FLASH_C2ACR) . . . . .	121
3.10.16	Flash memory CPU2 status register (FLASH_C2SR) . . . . .	122
3.10.17	Flash memory CPU2 control register (FLASH_C2CR) . . . . .	124
3.10.18	Secure flash memory start address register (FLASH_SFR) . . . . .	125

---

3.10.19	Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR) . . . . .	126
3.10.20	FLASH register map . . . . .	128
<b>4</b>	<b>Radio system . . . . .</b>	<b>130</b>
4.1	Introduction . . . . .	130
4.2	Main features . . . . .	130
4.3	Radio system functional description . . . . .	131
4.3.1	General description . . . . .	131
<b>5</b>	<b>Cyclic redundancy check calculation unit (CRC) . . . . .</b>	<b>132</b>
5.1	Introduction . . . . .	132
5.2	CRC main features . . . . .	132
5.3	CRC functional description . . . . .	133
5.3.1	CRC block diagram . . . . .	133
5.3.2	CRC internal signals . . . . .	133
5.3.3	CRC operation . . . . .	133
5.4	CRC registers . . . . .	135
5.4.1	CRC data register (CRC_DR) . . . . .	135
5.4.2	CRC independent data register (CRC_IDR) . . . . .	135
5.4.3	CRC control register (CRC_CR) . . . . .	136
5.4.4	CRC initial value (CRC_INIT) . . . . .	137
5.4.5	CRC polynomial (CRC_POL) . . . . .	137
5.4.6	CRC register map . . . . .	138
<b>6</b>	<b>Power control (PWR) . . . . .</b>	<b>139</b>
6.1	Power supplies . . . . .	139
6.1.1	Independent analog peripherals supply . . . . .	142
6.1.2	Independent USB transceivers supply . . . . .	143
6.1.3	Independent LCD supply (available only on STM32WB55xx) . . . . .	143
6.1.4	Battery backup domain . . . . .	144
6.1.5	Voltage regulator . . . . .	145
6.1.6	Dynamic voltage scaling management . . . . .	145
6.2	Power supply supervisor . . . . .	146
6.2.1	Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR) . . . . .	146
6.2.2	Programmable voltage detector (PVD) . . . . .	147

---

6.2.3	Peripheral voltage monitoring (PVM) . . . . .	148
6.3	CPU2 boot . . . . .	149
6.4	Low-power modes . . . . .	151
6.4.1	Run mode . . . . .	157
6.4.2	Low-power run mode (LP run) . . . . .	157
6.4.3	Entering Low-power mode . . . . .	158
6.4.4	Exiting Low-power mode . . . . .	158
6.4.5	Sleep mode . . . . .	160
6.4.6	Low-power sleep mode (LP sleep) . . . . .	161
6.4.7	Stop0 mode . . . . .	162
6.4.8	Stop1 mode . . . . .	164
6.4.9	Stop2 mode . . . . .	165
6.4.10	Standby mode . . . . .	167
6.4.11	Shutdown mode . . . . .	169
6.4.12	Auto wakeup from Low-power mode . . . . .	170
6.5	Real-time radio information . . . . .	171
6.6	PWR registers . . . . .	172
6.6.1	PWR control register 1 (PWR_CR1) . . . . .	172
6.6.2	PWR control register 2 (PWR_CR2) . . . . .	173
6.6.3	PWR control register 3 (PWR_CR3) . . . . .	174
6.6.4	PWR control register 4 (PWR_CR4) . . . . .	176
6.6.5	PWR status register 1 (PWR_SR1) . . . . .	177
6.6.6	PWR status register 2 (PWR_SR2) . . . . .	178
6.6.7	PWR status clear register (PWR_SCR) . . . . .	179
6.6.8	PWR control register 5 (PWR_CR5) . . . . .	180
6.6.9	PWR Port A pull-up control register (PWR_PUCRA) . . . . .	181
6.6.10	PWR Port A pull-down control register (PWR_PDCRA) . . . . .	182
6.6.11	PWR Port B pull-up control register (PWR_PUCRB) . . . . .	182
6.6.12	PWR Port B pull-down control register (PWR_PDCRB) . . . . .	183
6.6.13	PWR Port C pull-up control register (PWR_PUCRC) . . . . .	183
6.6.14	PWR Port C pull-down control register (PWR_PDCRC) . . . . .	184
6.6.15	PWR Port D pull-up control register (PWR_PUCRD) (STM32WB55xx only) . . . . .	184
6.6.16	PWR Port D pull-down control register (PWR_PDCRD) (STM32WB55xx only) . . . . .	185
6.6.17	PWR Port E pull-up control register (PWR_PUCRE) . . . . .	185
6.6.18	PWR Port E pull-down control register (PWR_PDCRE) . . . . .	186

---

6.6.19	PWR Port H pull-up control register (PWR_PUCRH) . . . . .	186
6.6.20	PWR Port H pull-down control register (PWR_PDCRH) . . . . .	187
6.6.21	PWR CPU2 control register 1 (PWR_C2CR1) . . . . .	187
6.6.22	PWR CPU2 control register 3 (PWR_C2CR3) . . . . .	189
6.6.23	PWR extended status and status clear register (PWR_EXTSCR) . .	190
6.6.24	PWR register map and reset value table . . . . .	192
<b>7</b>	<b>Peripherals interconnect matrix . . . . .</b>	<b>194</b>
7.1	Introduction . . . . .	194
7.2	Interconnect matrix implementation . . . . .	194
7.3	Connection summary . . . . .	194
7.4	Interconnection details . . . . .	195
7.4.1	From timer (TIM1/TIM2/TIM17) to timer (TIM1/TIM2) . . . . .	195
7.4.2	From timer (TIM1/TIM2) and EXTI to ADC (ADC1) . . . . .	196
7.4.3	From ADC (ADC1) to timer (TIM1) . . . . .	196
7.4.4	From HSE, LSE, LSI, MSI, MCO, RTC to timers (TIM2/TIM16/TIM17) . . . . .	196
7.4.5	From RTC, COMP1, COMP2 to low-power timers (LPTIM1/LPTIM2) .	197
7.4.6	From timer (TIM1/TIM2) to comparators (COMP1/COMP2) . . . .	197
7.4.7	From USB to timer (TIM2) . . . . .	198
7.4.8	From internal analog to ADC1 . . . . .	198
7.4.9	From comparators (COMP1/COMP2) to timers (TIM1/TIM2/TIM16/TIM17) . . . . .	198
7.4.10	From system errors to timers (TIM1/TIM16/TIM17) . . . . .	199
7.4.11	From timers (TIM16/TIM17) to IRTIM . . . . .	199
<b>8</b>	<b>Reset and clock control (RCC) . . . . .</b>	<b>200</b>
8.1	Reset . . . . .	200
8.1.1	Power reset . . . . .	200
8.1.2	System reset . . . . .	200
8.1.3	Backup domain reset . . . . .	202
8.2	Clocks . . . . .	202
8.2.1	HSE clock . . . . .	206
8.2.2	HSI16 clock . . . . .	207
8.2.3	MSI clock . . . . .	207
8.2.4	HSI48 clock . . . . .	208
8.2.5	PLLs . . . . .	209

---

8.2.6	LSE clock .....	209
8.2.7	LSI1 clock .....	210
8.2.8	LSI2 clock .....	211
8.2.9	System clock (SYSCLK) selection .....	211
8.2.10	Clock source frequency versus voltage scaling .....	211
8.2.11	Clock security system (CSS) on HSE .....	212
8.2.12	Clock security system on LSE (LSECSS) .....	212
8.2.13	LSI source selection .....	213
8.2.14	SMPS step-down converter clock .....	213
8.2.15	ADC clock .....	214
8.2.16	RTC clock .....	214
8.2.17	Timer clock .....	214
8.2.18	Watchdog clock .....	215
8.2.19	True RNG clock .....	215
8.2.20	Clock-out capability .....	215
8.2.21	Internal/external clock measurement with TIM16/TIM17 .....	216
8.2.22	Peripheral clocks enable .....	217
8.3	Low-power modes .....	219
8.4	RCC registers .....	221
8.4.1	RCC clock control register (RCC_CR) .....	221
8.4.2	RCC internal clock sources calibration register (RCC_ICSCR) .....	224
8.4.3	RCC clock configuration register (RCC_CFGR) .....	225
8.4.4	RCC PLL configuration register (RCC_PLLCFGR) .....	228
8.4.5	RCC PLLSAI1 configuration register (RCC_PLLSAI1CFGR) .....	231
8.4.6	RCC clock interrupt enable register (RCC_CIER) .....	233
8.4.7	RCC clock interrupt flag register (RCC_CIFR) .....	234
8.4.8	RCC clock interrupt clear register (RCC_CICR) .....	236
8.4.9	RCC SMPS step-down converter control register (RCC_SMPSCR) ..	237
8.4.10	RCC AHB1 peripheral reset register (RCC_AHB1RSTR) .....	238
8.4.11	RCC AHB2 peripheral reset register (RCC_AHB2RSTR) .....	239
8.4.12	RCC AHB3 and AHB4 peripheral reset register (RCC_AHB3RSTR) ..	240
8.4.13	RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1) .....	241
8.4.14	RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2) .....	243
8.4.15	RCC APB2 peripheral reset register (RCC_APB2RSTR) .....	243
8.4.16	RCC APB3 peripheral reset register (RCC_APB3RSTR) .....	244
8.4.17	RCC AHB1 peripheral clock enable register (RCC_AHB1ENR) .....	245
8.4.18	RCC AHB2 peripheral clock enable register (RCC_AHB2ENR) .....	246

8.4.19	RCC AHB3 and AHB4 peripheral clock enable register (RCC_AHB3ENR) .....	247
8.4.20	RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1) ...	248
8.4.21	RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2) ...	250
8.4.22	RCC APB2 peripheral clock enable register (RCC_APB2ENR) .....	250
8.4.23	RCC AHB1 peripheral clocks enable in Sleep modes register (RCC_AHB1SMENR) .....	251
8.4.24	RCC AHB2 peripheral clocks enable in Sleep modes register (RCC_AHB2SMENR) .....	252
8.4.25	RCC AHB3 and AHB4 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB3SMENR) .....	254
8.4.26	RCC APB1 peripheral clocks enable in Sleep mode register 1 (RCC_APB1SMENR1) .....	255
8.4.27	RCC APB1 peripheral clocks enable in Sleep mode register 2 (RCC_APB1SMENR2) .....	257
8.4.28	RCC APB2 peripheral clocks enable in Sleep mode register (RCC_APB2SMENR) .....	257
8.4.29	RCC peripherals independent clock configuration register (RCC_CCIPR) .....	259
8.4.30	RCC backup domain control register (RCC_BDCR) .....	260
8.4.31	RCC control/status register (RCC_CSR) .....	262
8.4.32	RCC clock recovery RC register (RCC_CRRCR) .....	264
8.4.33	RCC clock HSE register (RCC_HSECR) .....	265
8.4.34	RCC extended clock recovery register (RCC_EXTCFGR) .....	266
8.4.35	RCC CPU2 AHB1 peripheral clock enable register (RCC_C2AHB1ENR) .....	268
8.4.36	RCC CPU2 AHB2 peripheral clock enable register (RCC_C2AHB2ENR) .....	269
8.4.37	RCC CPU2 AHB3 and AHB4 peripheral clock enable register (RCC_C2AHB3ENR) .....	270
8.4.38	RCC CPU2 APB1 peripheral clock enable register 1 (RCC_C2APB1ENR1) .....	271
8.4.39	RCC CPU2 APB1 peripheral clock enable register 2 (RCC_C2APB1ENR2) .....	273
8.4.40	RCC CPU2 APB2 peripheral clock enable register (RCC_C2APB2ENR) .....	273
8.4.41	RCC CPU2 APB3 peripheral clock enable register (RCC_C2APB3ENR) .....	274
8.4.42	RCC CPU2 AHB1 peripheral clocks enable in Sleep modes register (RCC_C2AHB1SMENR) .....	275
8.4.43	RCC CPU2 AHB2 peripheral clocks enable in Sleep modes register (RCC_C2AHB2SMENR) .....	276

---

8.4.44	RCC CPU2 AHB3 and AHB4 peripheral clocks enable in Sleep mode register (RCC_C2AHB3SMENR) . . . . .	278
8.4.45	RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 1 (RCC_C2APB1SMENR1) . . . . .	279
8.4.46	RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 2 (RCC_C2APB1SMENR2) . . . . .	280
8.4.47	RCC CPU2 APB2 peripheral clocks enable in Sleep mode register (RCC_C2APB2SMENR) . . . . .	281
8.4.48	RCC CPU2 APB3 peripheral clock enable in Sleep mode register (RCC_C2APB3SMENR) . . . . .	282
8.4.49	RCC register map . . . . .	283
<b>9</b>	<b>General-purpose I/Os (GPIO) . . . . .</b>	<b>289</b>
9.1	Introduction . . . . .	289
9.2	GPIO main features . . . . .	289
9.3	GPIO implementation . . . . .	289
9.4	GPIO functional description . . . . .	290
9.4.1	General-purpose I/O (GPIO) . . . . .	292
9.4.2	I/O pin alternate function multiplexer and mapping . . . . .	292
9.4.3	I/O port control registers . . . . .	293
9.4.4	I/O port data registers . . . . .	293
9.4.5	I/O data bitwise handling . . . . .	293
9.4.6	GPIO locking mechanism . . . . .	294
9.4.7	I/O alternate function input/output . . . . .	294
9.4.8	External interrupt/wakeup lines . . . . .	294
9.4.9	Input configuration . . . . .	294
9.4.10	Output configuration . . . . .	295
9.4.11	Alternate function configuration . . . . .	296
9.4.12	Analog configuration . . . . .	296
9.4.13	Using the LSE oscillator pins as GPIOs . . . . .	297
9.4.14	Using the GPIO pins in the RTC supply domain . . . . .	297
9.4.15	Using PH3 as GPIO . . . . .	297
9.5	GPIO registers . . . . .	298
9.5.1	GPIO port mode register (GPIOx_MODER) (x = A to E and H) . . . . .	298
9.5.2	GPIO port output type register (GPIOx_OTYPER) (x = A to E and H) . . . . .	299
9.5.3	GPIO port output speed register (GPIOx_OSPEEDR) (x = A to E and H) . . . . .	299

9.5.4	GPIO port pull-up/pull-down register (GPIO <sub>x</sub> _PUPDR) (x = A to E and H) . . . . .	300
9.5.5	GPIO port input data register (GPIO <sub>x</sub> _IDR) (x = A to E and H) . . . . .	300
9.5.6	GPIO port output data register (GPIO <sub>x</sub> _ODR) (x = A to E and H) . . . . .	301
9.5.7	GPIO port bit set/reset register (GPIO <sub>x</sub> _BSRR) (x = A to E and H) . . . . .	301
9.5.8	GPIO port configuration lock register (GPIO <sub>x</sub> _LCKR) (x = A to E and H) . . . . .	302
9.5.9	GPIO alternate function low register (GPIO <sub>x</sub> _AFRL) (x = A to E and H) . . . . .	303
9.5.10	GPIO alternate function high register (GPIO <sub>x</sub> _AFRH) (x = A to E and H) . . . . .	304
9.5.11	GPIO port bit reset register (GPIO <sub>x</sub> _BRR) (x = A to E and H) . . . . .	305
9.5.12	GPIO register map . . . . .	306
<b>10</b>	<b>System configuration controller (SYSCFG) . . . . .</b>	<b>309</b>
10.1	SYSCFG main features . . . . .	309
10.2	SYSCFG registers . . . . .	309
10.2.1	SYSCFG memory remap register (SYSCFG_MEMRMP) . . . . .	309
10.2.2	SYSCFG configuration register 1 (SYSCFG_CFGR1) . . . . .	310
10.2.3	SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1) . . . . .	311
10.2.4	SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2) . . . . .	312
10.2.5	SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3) . . . . .	313
10.2.6	SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4) . . . . .	315
10.2.7	SYSCFG SRAM2 control and status register (SYSCFG_SCSR) . . . . .	316
10.2.8	SYSCFG configuration register 2 (SYSCFG_CFGR2) . . . . .	317
10.2.9	SYSCFG SRAM2 write protection register (SYSCFG_SWPR1) . . . . .	318
10.2.10	SYSCFG SRAM2 key register (SYSCFG_SKR) . . . . .	318
10.2.11	SYSCFG SRAM2 write protection register 2 (SYSCFG_SWPR2) . . . . .	318
10.2.12	SYSCFG CPU1 interrupt mask register 1 (SYSCFG_IMR1) . . . . .	319
10.2.13	SYSCFG CPU1 interrupt mask register 2 (SYSCFG_IMR2) . . . . .	319
10.2.14	SYSCFG CPU2 interrupt mask register 1 (SYSCFG_C2IMR1) . . . . .	320
10.2.15	SYSCFG CPU2 interrupt mask register 2 (SYSCFG_C2IMR2) . . . . .	321
10.2.16	SYSCFG secure IP control register (SYSCFG_SIPCR) . . . . .	321

---

10.2.17	SYSCFG register map	323
<b>11</b>	<b>Direct memory access controller (DMA)</b>	<b>325</b>
11.1	Introduction	325
11.2	DMA main features	325
11.3	DMA implementation	326
11.3.1	DMA1 and DMA2	326
11.3.2	DMA request mapping	326
11.4	DMA functional description	326
11.4.1	DMA block diagram	326
11.4.2	DMA pins and internal signals	328
11.4.3	DMA transfers	328
11.4.4	DMA arbitration	329
11.4.5	DMA channels	329
11.4.6	DMA data width, alignment and endianness	333
11.4.7	DMA error management	334
11.5	DMA interrupts	335
11.6	DMA registers	335
11.6.1	DMA interrupt status register (DMA_ISR)	335
11.6.2	DMA interrupt flag clear register (DMA_IFCR)	338
11.6.3	DMA channel x configuration register (DMA_CCRx)	339
11.6.4	DMA channel x number of data to transfer register (DMA_CNDTRx)	342
11.6.5	DMA channel x peripheral address register (DMA_CPARx)	342
11.6.6	DMA channel x memory address register (DMA_CMARx)	343
11.6.7	DMA register map	343
<b>12</b>	<b>DMA request multiplexer (DMAMUX)</b>	<b>346</b>
12.1	Introduction	346
12.2	DMAMUX main features	347
12.3	DMAMUX implementation	347
12.3.1	DMAMUX instantiation	347
12.3.2	DMAMUX mapping	347
12.4	DMAMUX functional description	350
12.4.1	DMAMUX block diagram	350
12.4.2	DMAMUX signals	351
12.4.3	DMAMUX channels	351

---

12.4.4	DMAMUX request line multiplexer . . . . .	351
12.4.5	DMAMUX request generator . . . . .	354
12.5	DMAMUX interrupts . . . . .	355
12.6	DMAMUX registers . . . . .	356
12.6.1	DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR) . . . . .	356
12.6.2	DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR) . . . . .	357
12.6.3	DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR) . . . . .	357
12.6.4	DMAMUX request generator channel x configuration register (DMAMUX_RGxCR) . . . . .	358
12.6.5	DMAMUX request generator interrupt status register (DMAMUX_RGSR) . . . . .	359
12.6.6	DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR) . . . . .	359
12.6.7	DMAMUX register map . . . . .	360
<b>13</b>	<b>Nested vectored interrupt controller (NVIC) . . . . .</b>	<b>362</b>
13.1	NVIC main features . . . . .	362
13.2	NVIC implementation . . . . .	362
13.3	Interrupt block diagram . . . . .	362
13.4	Interrupt and exception vectors . . . . .	363
13.5	Interrupt list . . . . .	369
<b>14</b>	<b>Extended interrupt and event controller (EXTI) . . . . .</b>	<b>371</b>
14.1	EXTI main features . . . . .	371
14.2	EXTI implementation . . . . .	371
14.3	EXTI block diagram . . . . .	372
14.3.1	EXTI connections between peripherals and CPU . . . . .	373
14.4	EXTI functional description . . . . .	374
14.4.1	EXTI configurable event input wakeup . . . . .	374
14.4.2	EXTI direct event input wakeup . . . . .	376
14.5	EXTI functional behavior . . . . .	376
14.6	EXTI registers . . . . .	378
14.6.1	EXTI rising trigger selection register (EXTI_RTSR1) . . . . .	378
14.6.2	EXTI falling trigger selection register (EXTI_FTSR1) . . . . .	379

---

14.6.3	EXTI software interrupt event register (EXTI_SWIER1) . . . . .	379
14.6.4	EXTI pending register (EXTI_PR1) . . . . .	380
14.6.5	EXTI rising trigger selection register (EXTI_RTSR2) . . . . .	380
14.6.6	EXTI falling trigger selection register (EXTI_FTSR2) . . . . .	381
14.6.7	EXTI software interrupt event register (EXTI_SWIER2) . . . . .	382
14.6.8	EXTI pending register (EXTI_PR2) . . . . .	383
14.6.9	EXTI CPU wakeup with interrupt mask register (EXTI_IMR1) . . . . .	383
14.6.10	EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR1) . . . . .	384
14.6.11	EXTI CPU wakeup with event mask register (EXTI_EMR1) . . . . .	384
14.6.12	EXTI CPU2 wakeup with event mask register (EXTI_C2EMR1) . . . . .	385
14.6.13	EXTI CPU wakeup with interrupt mask register (EXTI_IMR2) . . . . .	385
14.6.14	EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR2) . . . . .	386
14.6.15	EXTI CPU wakeup with event mask register (EXTI_EMR2) . . . . .	386
14.6.16	EXTI CPU2 wakeup with event mask register (EXTI_C2EMR2) . . . . .	387
14.6.17	EXTI register map . . . . .	388
<b>15</b>	<b>Quad-SPI interface (QUADSPI) . . . . .</b>	<b>390</b>
15.1	Introduction . . . . .	390
15.2	QUADSPI main features . . . . .	390
15.3	QUADSPI functional description . . . . .	390
15.3.1	QUADSPI block diagram . . . . .	390
15.3.2	QUADSPI pins . . . . .	391
15.3.3	QUADSPI command sequence . . . . .	391
15.3.4	QUADSPI signal interface protocol modes . . . . .	393
15.3.5	QUADSPI indirect mode . . . . .	395
15.3.6	QUADSPI automatic status-polling mode . . . . .	396
15.3.7	QUADSPI memory-mapped mode . . . . .	397
15.3.8	QUADSPI flash memory configuration . . . . .	398
15.3.9	QUADSPI delayed data sampling . . . . .	398
15.3.10	QUADSPI configuration . . . . .	398
15.3.11	QUADSPI use . . . . .	399
15.3.12	Sending the instruction only once . . . . .	401
15.3.13	QUADSPI error management . . . . .	401
15.3.14	QUADSPI busy bit and abort functionality . . . . .	401
15.3.15	NCS behavior . . . . .	401
15.4	QUADSPI interrupts . . . . .	403
15.5	QUADSPI registers . . . . .	404

---

15.5.1	QUADSPI control register (QUADSPI_CR) . . . . .	404
15.5.2	QUADSPI device configuration register (QUADSPI_DCR) . . . . .	406
15.5.3	QUADSPI status register (QUADSPI_SR) . . . . .	407
15.5.4	QUADSPI flag clear register (QUADSPI_FCR) . . . . .	408
15.5.5	QUADSPI data length register (QUADSPI_DLR) . . . . .	409
15.5.6	QUADSPI communication configuration register (QUADSPI_CCR) . .	409
15.5.7	QUADSPI address register (QUADSPI_AR) . . . . .	411
15.5.8	QUADSPI alternate-byte register (QUADSPI_ABR) . . . . .	411
15.5.9	QUADSPI data register (QUADSPI_DR) . . . . .	412
15.5.10	QUADSPI polling status mask register (QUADSPI_PSMKR) . . . .	412
15.5.11	QUADSPI polling status match register (QUADSPI_PSMAR) . . . .	413
15.5.12	QUADSPI polling interval register (QUADSPI_PIR) . . . . .	413
15.5.13	QUADSPI low-power timeout register (QUADSPI_LPTR) . . . . .	414
15.5.14	QUADSPI register map . . . . .	414
<b>16</b>	<b>Analog-to-digital converter (ADC) . . . . .</b>	<b>416</b>
16.1	Introduction . . . . .	416
16.2	ADC main features . . . . .	416
16.3	ADC implementation . . . . .	417
16.4	ADC functional description . . . . .	418
16.4.1	ADC block diagram . . . . .	418
16.4.2	ADC pins and internal signals . . . . .	419
16.4.3	ADC clocks . . . . .	420
16.4.4	ADC1 connectivity . . . . .	422
16.4.5	Slave AHB interface . . . . .	423
16.4.6	ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN) . . . . .	423
16.4.7	Single-ended and differential input channels . . . . .	424
16.4.8	Calibration (ADCAL, ADCALDIF, ADC_CALFACT) . . . . .	424
16.4.9	ADC on-off control (ADEN, ADDIS, ADRDY) . . . . .	427
16.4.10	Constraints when writing the ADC control bits . . . . .	428
16.4.11	Channel selection (SQRx, JSQRx) . . . . .	429
16.4.12	Channel-wise programmable sampling time (SMPR1, SMPR2) . . .	430
16.4.13	Single conversion mode (CONT = 0) . . . . .	430
16.4.14	Continuous conversion mode (CONT = 1) . . . . .	431
16.4.15	Starting conversions (ADSTART, JADSTART) . . . . .	432
16.4.16	ADC timing . . . . .	433

---

16.4.17	Stopping an ongoing conversion (ADSTP, JADSTP) . . . . .	433
16.4.18	Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) . . . . .	435
16.4.19	Injected channel management . . . . .	437
16.4.20	Discontinuous mode (DISCEN, DISCNUM, JDISCEN) . . . . .	438
16.4.21	Queue of context for injected conversions . . . . .	439
16.4.22	Programmable resolution (RES) - Fast conversion mode . . . . .	447
16.4.23	End of conversion, end of sampling phase (EOC, JEOC, EOSMP) . . . . .	448
16.4.24	End of conversion sequence (EOS, JEOS) . . . . .	448
16.4.25	Timing diagrams example (single/continuous modes, hardware/software triggers) . . . . .	449
16.4.26	Data management . . . . .	451
16.4.27	Dynamic low-power features . . . . .	456
16.4.28	Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_LTx, AWD_LTx, AWDx) . . . . .	461
16.4.29	Oversampler . . . . .	465
16.4.30	Temperature sensor . . . . .	470
16.4.31	VBAT supply monitoring . . . . .	472
16.4.32	Monitoring the internal voltage reference . . . . .	472
16.5	ADC in low-power mode . . . . .	474
16.6	ADC interrupts . . . . .	475
16.7	ADC registers . . . . .	476
16.7.1	ADC interrupt and status register (ADC_ISR) . . . . .	476
16.7.2	ADC interrupt enable register (ADC_IER) . . . . .	478
16.7.3	ADC control register (ADC_CR) . . . . .	480
16.7.4	ADC configuration register (ADC_CFGR) . . . . .	483
16.7.5	ADC configuration register 2 (ADC_CFGR2) . . . . .	487
16.7.6	ADC sample time register 1 (ADC_SMPR1) . . . . .	488
16.7.7	ADC sample time register 2 (ADC_SMPR2) . . . . .	489
16.7.8	ADC watchdog threshold register 1 (ADC_TR1) . . . . .	490
16.7.9	ADC watchdog threshold register 2 (ADC_TR2) . . . . .	490
16.7.10	ADC watchdog threshold register 3 (ADC_TR3) . . . . .	491
16.7.11	ADC regular sequence register 1 (ADC_SQR1) . . . . .	492
16.7.12	ADC regular sequence register 2 (ADC_SQR2) . . . . .	493
16.7.13	ADC regular sequence register 3 (ADC_SQR3) . . . . .	494
16.7.14	ADC regular sequence register 4 (ADC_SQR4) . . . . .	495
16.7.15	ADC regular data register (ADC_DR) . . . . .	495
16.7.16	ADC injected sequence register (ADC_JSQR) . . . . .	496

---

16.7.17	ADC offset y register (ADC_OFRy) . . . . .	498
16.7.18	ADC injected channel y data register (ADC_JDRy) . . . . .	499
16.7.19	ADC analog watchdog 2 configuration register (ADC_AWD2CR) . . . . .	499
16.7.20	ADC analog watchdog 3 configuration register (ADC_AWD3CR) . . . . .	500
16.7.21	ADC differential mode selection register (ADC_DIFSEL) . . . . .	500
16.7.22	ADC calibration factors (ADC_CALFACT) . . . . .	501
16.8	ADC common registers . . . . .	501
16.8.1	ADC common status register (ADC_CSR) . . . . .	501
16.8.2	ADC common control register (ADC_CCR) . . . . .	502
16.9	ADC register map . . . . .	504
<b>17</b>	<b>Voltage reference buffer (VREFBUF) . . . . .</b>	<b>507</b>
17.1	Introduction . . . . .	507
17.2	VREFBUF implementation . . . . .	507
17.3	VREFBUF functional description . . . . .	507
17.4	VREFBUF trimming . . . . .	508
17.5	VREFBUF registers . . . . .	508
17.5.1	VREFBUF control and status register (VREFBUF_CSR) . . . . .	508
17.5.2	VREFBUF calibration control register (VREFBUF_CCR) . . . . .	509
17.5.3	VREFBUF register map . . . . .	509
<b>18</b>	<b>Comparator (COMP) . . . . .</b>	<b>511</b>
18.1	Introduction . . . . .	511
18.2	COMP main features . . . . .	511
18.3	COMP functional description . . . . .	512
18.3.1	COMP block diagram . . . . .	512
18.3.2	COMP pins and internal signals . . . . .	512
18.3.3	COMP reset and clocks . . . . .	514
18.3.4	Comparator LOCK mechanism . . . . .	514
18.3.5	Window comparator . . . . .	514
18.3.6	Hysteresis . . . . .	515
18.3.7	Comparator output blanking function . . . . .	516
18.3.8	COMP power and speed modes . . . . .	516
18.4	COMP low-power modes . . . . .	517
18.5	COMP interrupts . . . . .	517
18.6	COMP registers . . . . .	518

---

18.6.1	Comparator 1 control and status register (COMP1_CSR) . . . . .	518
18.6.2	Comparator 2 control and status register (COMP2_CSR) . . . . .	520
18.6.3	COMP register map . . . . .	523
<b>19</b>	<b>Liquid crystal display controller (LCD) . . . . .</b>	<b>524</b>
19.1	LCD introduction . . . . .	524
19.2	LCD main features . . . . .	524
19.3	LCD functional description . . . . .	526
19.3.1	General description . . . . .	526
19.3.2	Frequency generator . . . . .	527
19.3.3	Common driver . . . . .	528
19.3.4	Segment driver . . . . .	531
19.3.5	Voltage generator and contrast control . . . . .	535
19.3.6	Double buffer memory . . . . .	538
19.3.7	COM and SEG multiplexing . . . . .	539
19.3.8	Flowchart . . . . .	545
19.4	LCD low-power modes . . . . .	546
19.5	LCD interrupts . . . . .	546
19.6	LCD registers . . . . .	547
19.6.1	LCD control register (LCD_CR) . . . . .	547
19.6.2	LCD frame control register (LCD_FCR) . . . . .	548
19.6.3	LCD status register (LCD_SR) . . . . .	551
19.6.4	LCD clear register (LCD_CLR) . . . . .	552
19.6.5	LCD display memory (LCD_RAM) . . . . .	552
19.6.6	LCD register map . . . . .	553
<b>20</b>	<b>Touch sensing controller (TSC) . . . . .</b>	<b>555</b>
20.1	Introduction . . . . .	555
20.2	TSC main features . . . . .	555
20.3	TSC functional description . . . . .	556
20.3.1	TSC block diagram . . . . .	556
20.3.2	Surface charge transfer acquisition overview . . . . .	556
20.3.3	Reset and clocks . . . . .	558
20.3.4	Charge transfer acquisition sequence . . . . .	559
20.3.5	Spread spectrum feature . . . . .	560
20.3.6	Max count error . . . . .	560

---

20.3.7	Sampling capacitor I/O and channel I/O mode selection . . . . .	561
20.3.8	Acquisition mode . . . . .	562
20.3.9	I/O hysteresis and analog switch control . . . . .	562
20.4	TSC low-power modes . . . . .	563
20.5	TSC interrupts . . . . .	563
20.6	TSC registers . . . . .	564
20.6.1	TSC control register (TSC_CR) . . . . .	564
20.6.2	TSC interrupt enable register (TSC_IER) . . . . .	566
20.6.3	TSC interrupt clear register (TSC_ICR) . . . . .	567
20.6.4	TSC interrupt status register (TSC_ISR) . . . . .	568
20.6.5	TSC I/O hysteresis control register (TSC_IOHCR) . . . . .	568
20.6.6	TSC I/O analog switch control register (TSC_IOASCR) . . . . .	569
20.6.7	TSC I/O sampling control register (TSC_IOSCR) . . . . .	569
20.6.8	TSC I/O channel control register (TSC_IOCCR) . . . . .	570
20.6.9	TSC I/O group control status register (TSC_IOGCSR) . . . . .	570
20.6.10	TSC I/O group x counter register (TSC_IOGxCR) . . . . .	571
20.6.11	TSC register map . . . . .	572
<b>21</b>	<b>True random number generator (RNG) . . . . .</b>	<b>574</b>
21.1	Introduction . . . . .	574
21.2	RNG main features . . . . .	574
21.3	RNG functional description . . . . .	575
21.3.1	RNG block diagram . . . . .	575
21.3.2	RNG internal signals . . . . .	575
21.3.3	Random number generation . . . . .	576
21.3.4	RNG initialization . . . . .	578
21.3.5	RNG operation . . . . .	579
21.3.6	RNG clocking . . . . .	580
21.3.7	Error management . . . . .	580
21.3.8	RNG low-power usage . . . . .	581
21.4	RNG interrupts . . . . .	582
21.5	RNG processing time . . . . .	582
21.6	RNG entropy source validation . . . . .	582
21.6.1	Introduction . . . . .	582
21.6.2	Validation conditions . . . . .	582

---

21.7	RNG registers . . . . .	583
21.7.1	RNG control register (RNG_CR) . . . . .	583
21.7.2	RNG status register (RNG_SR) . . . . .	584
21.7.3	RNG data register (RNG_DR) . . . . .	585
21.7.4	RNG register map . . . . .	585
<b>22</b>	<b>AES hardware accelerator (AES) . . . . .</b>	<b>586</b>
22.1	Introduction . . . . .	586
22.2	AES main features . . . . .	586
22.3	AES implementation . . . . .	587
22.4	AES functional description . . . . .	587
22.4.1	AES block diagram . . . . .	587
22.4.2	AES internal signals . . . . .	587
22.4.3	AES cryptographic core . . . . .	588
22.4.4	AES procedure to perform a cipher operation . . . . .	593
22.4.5	AES decryption round key preparation . . . . .	596
22.4.6	AES ciphertext stealing and data padding . . . . .	597
22.4.7	AES task suspend and resume . . . . .	597
22.4.8	AES basic chaining modes (ECB, CBC) . . . . .	598
22.4.9	AES counter (CTR) mode . . . . .	603
22.4.10	AES Galois/counter mode (GCM) . . . . .	605
22.4.11	AES Galois message authentication code (GMAC) . . . . .	610
22.4.12	AES counter with CBC-MAC (CCM) . . . . .	612
22.4.13	AES data registers and data swapping . . . . .	617
22.4.14	AES key registers . . . . .	619
22.4.15	AES initialization vector registers . . . . .	619
22.4.16	AES DMA interface . . . . .	620
22.4.17	AES error management . . . . .	621
22.5	AES interrupts . . . . .	622
22.6	AES processing latency . . . . .	622
22.7	AES registers . . . . .	623
22.7.1	AES control register (AES_CR) . . . . .	623
22.7.2	AES status register (AES_SR) . . . . .	626
22.7.3	AES data input register (AES_DINR) . . . . .	627
22.7.4	AES data output register (AES_DOUTR) . . . . .	627
22.7.5	AES key register 0 (AES_KEYR0) . . . . .	628

---

22.7.6	AES key register 1 (AES_KEYR1) . . . . .	629
22.7.7	AES key register 2 (AES_KEYR2) . . . . .	629
22.7.8	AES key register 3 (AES_KEYR3) . . . . .	629
22.7.9	AES initialization vector register 0 (AES_IVR0) . . . . .	630
22.7.10	AES initialization vector register 1 (AES_IVR1) . . . . .	630
22.7.11	AES initialization vector register 2 (AES_IVR2) . . . . .	630
22.7.12	AES initialization vector register 3 (AES_IVR3) . . . . .	631
22.7.13	AES key register 4 (AES_KEYR4) . . . . .	631
22.7.14	AES key register 5 (AES_KEYR5) . . . . .	631
22.7.15	AES key register 6 (AES_KEYR6) . . . . .	632
22.7.16	AES key register 7 (AES_KEYR7) . . . . .	632
22.7.17	AES suspend registers (AES_SUSPxR) . . . . .	632
22.7.18	AES register map . . . . .	633
<b>23</b>	<b>Public key accelerator (PKA) . . . . .</b>	<b>635</b>
23.1	Introduction . . . . .	635
23.2	PKA main features . . . . .	635
23.3	PKA functional description . . . . .	635
23.3.1	PKA block diagram . . . . .	635
23.3.2	PKA internal signals . . . . .	636
23.3.3	PKA reset and clocks . . . . .	636
23.3.4	PKA public key acceleration . . . . .	636
23.3.5	Typical applications for PKA . . . . .	638
23.3.6	PKA procedure to perform an operation . . . . .	640
23.3.7	PKA error management . . . . .	641
23.4	PKA operating modes . . . . .	641
23.4.1	Introduction . . . . .	641
23.4.2	Montgomery parameter computation . . . . .	642
23.4.3	Modular addition . . . . .	643
23.4.4	Modular subtraction . . . . .	643
23.4.5	Modular and Montgomery multiplication . . . . .	643
23.4.6	Modular exponentiation . . . . .	644
23.4.7	Modular inversion . . . . .	645
23.4.8	Modular reduction . . . . .	646
23.4.9	Arithmetic addition . . . . .	646
23.4.10	Arithmetic subtraction . . . . .	646
23.4.11	Arithmetic multiplication . . . . .	647

---

23.4.12	Arithmetic comparison . . . . .	647
23.4.13	RSA CRT exponentiation . . . . .	647
23.4.14	Point on elliptic curve Fp check . . . . .	648
23.4.15	ECC Fp scalar multiplication . . . . .	649
23.4.16	ECDSA sign . . . . .	650
23.4.17	ECDSA verification . . . . .	652
23.5	Example of configurations and processing times . . . . .	653
23.5.1	Supported elliptic curves . . . . .	653
23.5.2	Computation times . . . . .	655
23.6	PKA interrupts . . . . .	656
23.7	PKA registers . . . . .	657
23.7.1	PKA control register (PKA_CR) . . . . .	657
23.7.2	PKA status register (PKA_SR) . . . . .	658
23.7.3	PKA clear flag register (PKA_CLRFR) . . . . .	659
23.7.4	PKA RAM . . . . .	659
23.7.5	PKA register map . . . . .	660
<b>24</b>	<b>Advanced-control timer (TIM1) . . . . .</b>	<b>661</b>
24.1	TIM1 introduction . . . . .	661
24.2	TIM1 main features . . . . .	662
24.3	TIM1 functional description . . . . .	664
24.3.1	Time-base unit . . . . .	664
24.3.2	Counter modes . . . . .	666
24.3.3	Repetition counter . . . . .	677
24.3.4	External trigger input . . . . .	679
24.3.5	Clock selection . . . . .	680
24.3.6	Capture/compare channels . . . . .	684
24.3.7	Input capture mode . . . . .	686
24.3.8	PWM input mode . . . . .	687
24.3.9	Forced output mode . . . . .	688
24.3.10	Output compare mode . . . . .	689
24.3.11	PWM mode . . . . .	690
24.3.12	Asymmetric PWM mode . . . . .	693
24.3.13	Combined PWM mode . . . . .	694
24.3.14	Combined 3-phase PWM mode . . . . .	695
24.3.15	Complementary outputs and dead-time insertion . . . . .	696

24.3.16	Using the break function . . . . .	698
24.3.17	Bidirectional break inputs . . . . .	704
24.3.18	Clearing the OCxREF signal on an external event . . . . .	706
24.3.19	6-step PWM generation . . . . .	707
24.3.20	One-pulse mode . . . . .	708
24.3.21	Retriggerable one pulse mode . . . . .	709
24.3.22	Encoder interface mode . . . . .	710
24.3.23	UIF bit remapping . . . . .	712
24.3.24	Timer input XOR function . . . . .	713
24.3.25	Interfacing with Hall sensors . . . . .	713
24.3.26	Timer synchronization . . . . .	716
24.3.27	ADC synchronization . . . . .	720
24.3.28	DMA burst mode . . . . .	720
24.3.29	Debug mode . . . . .	721
24.4	TIM1 registers . . . . .	722
24.4.1	TIM1 control register 1 (TIM1_CR1) . . . . .	722
24.4.2	TIM1 control register 2 (TIM1_CR2) . . . . .	723
24.4.3	TIM1 slave mode control register (TIM1_SMCR) . . . . .	726
24.4.4	TIM1 DMA/interrupt enable register (TIM1_DIER) . . . . .	728
24.4.5	TIM1 status register (TIM1_SR) . . . . .	730
24.4.6	TIM1 event generation register (TIM1_EGR) . . . . .	732
24.4.7	TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1) . . . . .	733
24.4.8	TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1) . . . . .	734
24.4.9	TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2) . . . . .	737
24.4.10	TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2) . . . . .	738
24.4.11	TIM1 capture/compare enable register (TIM1_CCER) . . . . .	740
24.4.12	TIM1 counter (TIM1_CNT) . . . . .	743
24.4.13	TIM1 prescaler (TIM1_PSC) . . . . .	743
24.4.14	TIM1 auto-reload register (TIM1_ARR) . . . . .	743
24.4.15	TIM1 repetition counter register (TIM1_RCR) . . . . .	744
24.4.16	TIM1 capture/compare register 1 (TIM1_CCR1) . . . . .	744

---

24.4.17	TIM1 capture/compare register 2 (TIM1_CCR2) .....	745
24.4.18	TIM1 capture/compare register 3 (TIM1_CCR3) .....	745
24.4.19	TIM1 capture/compare register 4 (TIM1_CCR4) .....	746
24.4.20	TIM1 break and dead-time register (TIM1_BDTR) .....	746
24.4.21	TIM1 DMA control register (TIM1_DCR) .....	750
24.4.22	TIM1 DMA address for full transfer (TIM1_DMAR) .....	751
24.4.23	TIM1 option register 1 (TIM1_OR1) .....	752
24.4.24	TIM1 capture/compare mode register 3 (TIM1_CCMR3) .....	752
24.4.25	TIM1 capture/compare register 5 (TIM1_CCR5) .....	753
24.4.26	TIM1 capture/compare register 6 (TIM1_CCR6) .....	754
24.4.27	TIM1 alternate function option register 1 (TIM1_AF1) .....	755
24.4.28	TIM1 Alternate function register 2 (TIM1_AF2) .....	756
24.4.29	TIM1 timer input selection register (TIM1_TISEL) .....	758
24.4.30	TIM1 register map .....	759
<b>25</b>	<b>General-purpose timer (TIM2) .....</b>	<b>762</b>
25.1	TIM2 introduction .....	762
25.2	TIM2 main features .....	762
25.3	TIM2 functional description .....	764
25.3.1	Time-base unit .....	764
25.3.2	Counter modes .....	766
25.3.3	Clock selection .....	776
25.3.4	Capture/Compare channels .....	780
25.3.5	Input capture mode .....	782
25.3.6	PWM input mode .....	783
25.3.7	Forced output mode .....	784
25.3.8	Output compare mode .....	784
25.3.9	PWM mode .....	785
25.3.10	Asymmetric PWM mode .....	789
25.3.11	Combined PWM mode .....	789

25.3.12	Clearing the OC <sub>x</sub> REF signal on an external event	790
25.3.13	One-pulse mode	792
25.3.14	Retriggerable one pulse mode	793
25.3.15	Encoder interface mode	794
25.3.16	UIF bit remapping	796
25.3.17	Timer input XOR function	796
25.3.18	Timers and external trigger synchronization	797
25.3.19	Timer synchronization	800
25.3.20	DMA burst mode	804
25.3.21	Debug mode	805
25.4	TIM2 registers	806
25.4.1	TIM2 control register 1 (TIM2_CR1)	806
25.4.2	TIM2 control register 2 (TIM2_CR2)	807
25.4.3	TIM2 slave mode control register (TIM2_SMCR)	809
25.4.4	TIM2 DMA/Interrupt enable register (TIM2_DIER)	812
25.4.5	TIM2 status register (TIM2_SR)	813
25.4.6	TIM2 event generation register (TIM2_EGR)	815
25.4.7	TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1)	816
25.4.8	TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1)	818
25.4.9	TIM2 capture/compare mode register 2 [alternate] (TIM2_CCMR2)	820
25.4.10	TIM2 capture/compare mode register 2 [alternate] (TIM2_CCMR2)	821
25.4.11	TIM2 capture/compare enable register (TIM2_CCER)	822
25.4.12	TIM2 counter [alternate] (TIM2_CNT)	823
25.4.13	TIM2 counter [alternate] (TIM2_CNT)	824
25.4.14	TIM2 prescaler (TIM2_PSC)	824
25.4.15	TIM2 auto-reload register (TIM2_ARR)	825
25.4.16	TIM2 capture/compare register 1 (TIM2_CCR1)	825
25.4.17	TIM2 capture/compare register 2 (TIM2_CCR2)	825
25.4.18	TIM2 capture/compare register 3 (TIM2_CCR3)	826
25.4.19	TIM2 capture/compare register 4 (TIM2_CCR4)	826
25.4.20	TIM2 DMA control register (TIM2_DCR)	827
25.4.21	TIM2 DMA address for full transfer (TIM2_DMAR)	828
25.4.22	TIM2 option register 1 (TIM2_OR1)	828
25.4.23	TIM2 alternate function option register 1 (TIM2_AF1)	828
25.4.24	TIM2 timer input selection register (TIM2_TISEL)	829
25.4.25	TIMx register map	830

<b>26</b>	<b>General-purpose timers (TIM16/TIM17) . . . . .</b>	<b>833</b>
26.1	TIM16/TIM17 introduction . . . . .	833
26.2	TIM16/TIM17 main features . . . . .	833
26.3	TIM16/TIM17 functional description . . . . .	835
26.3.1	Time-base unit . . . . .	835
26.3.2	Counter modes . . . . .	837
26.3.3	Repetition counter . . . . .	841
26.3.4	Clock selection . . . . .	842
26.3.5	Capture/compare channels . . . . .	844
26.3.6	Input capture mode . . . . .	846
26.3.7	Forced output mode . . . . .	847
26.3.8	Output compare mode . . . . .	847
26.3.9	PWM mode . . . . .	849
26.3.10	Complementary outputs and dead-time insertion . . . . .	850
26.3.11	Using the break function . . . . .	852
26.3.12	Bidirectional break inputs . . . . .	855
26.3.13	6-step PWM generation . . . . .	856
26.3.14	One-pulse mode . . . . .	858
26.3.15	UIF bit remapping . . . . .	859
26.3.16	Slave mode – combined reset + trigger mode . . . . .	859
26.3.17	DMA burst mode . . . . .	859
26.3.18	Using timer output as trigger for other timers (TIM16/TIM17) . . . . .	860
26.3.19	Debug mode . . . . .	861
26.4	TIM16/TIM17 registers . . . . .	862
26.4.1	TIMx control register 1 (TIMx_CR1)(x = 16 to 17) . . . . .	862
26.4.2	TIMx control register 2 (TIMx_CR2)(x = 16 to 17) . . . . .	863
26.4.3	TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) . . . . .	864
26.4.4	TIMx status register (TIMx_SR)(x = 16 to 17) . . . . .	865
26.4.5	TIMx event generation register (TIMx_EGR)(x = 16 to 17) . . . . .	866
26.4.6	TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) . . . . .	867
26.4.7	TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) . . . . .	868
26.4.8	TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) . . . . .	870
26.4.9	TIMx counter (TIMx_CNT)(x = 16 to 17) . . . . .	872
26.4.10	TIMx prescaler (TIMx_PSC)(x = 16 to 17) . . . . .	873
26.4.11	TIMx auto-reload register (TIMx_ARR)(x = 16 to 17) . . . . .	873

---

26.4.12	TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) . . . . .	874
26.4.13	TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) . . . . .	874
26.4.14	TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) . . . . .	875
26.4.15	TIMx DMA control register (TIMx_DCR)(x = 16 to 17) . . . . .	877
26.4.16	TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) . . . . .	878
26.4.17	TIM16 option register 1 (TIM16_OR1) . . . . .	879
26.4.18	TIM16 alternate function register 1 (TIM16_AF1) . . . . .	879
26.4.19	TIM16 input selection register (TIM16_TISEL) . . . . .	880
26.4.20	TIM17 option register 1 (TIM17_OR1) . . . . .	880
26.4.21	TIM17 alternate function register 1 (TIM17_AF1) . . . . .	881
26.4.22	TIM17 input selection register (TIM17_TISEL) . . . . .	882
26.4.23	TIM16/TIM17 register map . . . . .	883
<b>27</b>	<b>Low-power timer (LPTIM) . . . . .</b>	<b>885</b>
27.1	Introduction . . . . .	885
27.2	LPTIM main features . . . . .	885
27.3	LPTIM implementation . . . . .	886
27.4	LPTIM functional description . . . . .	887
27.4.1	LPTIM block diagram . . . . .	887
27.4.2	LPTIM trigger mapping . . . . .	887
27.4.3	LPTIM reset and clocks . . . . .	888
27.4.4	Glitch filter . . . . .	888
27.4.5	Prescaler . . . . .	889
27.4.6	Trigger multiplexer . . . . .	890
27.4.7	Operating mode . . . . .	890
27.4.8	Timeout function . . . . .	892
27.4.9	Waveform generation . . . . .	892
27.4.10	Register update . . . . .	893
27.4.11	Counter mode . . . . .	894
27.4.12	Timer enable . . . . .	894
27.4.13	Timer counter reset . . . . .	895
27.4.14	Encoder mode . . . . .	895
27.4.15	Debug mode . . . . .	897
27.5	LPTIM low-power modes . . . . .	897
27.6	LPTIM interrupts . . . . .	898
27.7	LPTIM registers . . . . .	898

---

27.7.1	LPTIM interrupt and status register (LPTIM_ISR) . . . . .	899
27.7.2	LPTIM interrupt clear register (LPTIM_ICR) . . . . .	900
27.7.3	LPTIM interrupt enable register (LPTIM_IER) . . . . .	900
27.7.4	LPTIM configuration register (LPTIM_CFGR) . . . . .	901
27.7.5	LPTIM control register (LPTIM_CR) . . . . .	904
27.7.6	LPTIM compare register (LPTIM_CMP) . . . . .	906
27.7.7	LPTIM autoreload register (LPTIM_ARR) . . . . .	906
27.7.8	LPTIM counter register (LPTIM_CNT) . . . . .	907
27.7.9	LPTIM1 option register (LPTIM1_OR) . . . . .	907
27.7.10	LPTIM2 option register (LPTIM2_OR) . . . . .	908
27.7.11	LPTIM register map . . . . .	909
<b>28</b>	<b>Infrared interface (IRTIM) . . . . .</b>	<b>910</b>
<b>29</b>	<b>Real-time clock (RTC) . . . . .</b>	<b>911</b>
29.1	Introduction . . . . .	911
29.2	RTC main features . . . . .	912
29.3	RTC implementation . . . . .	912
29.4	RTC functional description . . . . .	913
29.4.1	RTC block diagram . . . . .	913
29.4.2	Clock and prescalers . . . . .	914
29.4.3	Real-time clock and calendar . . . . .	914
29.4.4	Programmable alarms . . . . .	915
29.4.5	Periodic auto-wake-up . . . . .	915
29.4.6	RTC initialization and configuration . . . . .	916
29.4.7	Reading the calendar . . . . .	917
29.4.8	Resetting the RTC . . . . .	918
29.4.9	RTC synchronization . . . . .	919
29.4.10	RTC reference clock detection . . . . .	919
29.4.11	RTC smooth digital calibration . . . . .	920
29.4.12	Time-stamp function . . . . .	922
29.4.13	Tamper detection . . . . .	923
29.4.14	Calibration clock output . . . . .	925
29.4.15	Alarm output . . . . .	925
29.5	RTC low-power modes . . . . .	926
29.6	RTC interrupts . . . . .	926

---

29.7	RTC registers	927
29.7.1	RTC time register (RTC_TR)	927
29.7.2	RTC date register (RTC_DR)	928
29.7.3	RTC control register (RTC_CR)	929
29.7.4	RTC initialization and status register (RTC_ISR)	932
29.7.5	RTC prescaler register (RTC_PRER)	935
29.7.6	RTC wake-up timer register (RTC_WUTR)	936
29.7.7	RTC alarm A register (RTC_ALRMAR)	937
29.7.8	RTC alarm B register (RTC_ALRMBR)	938
29.7.9	RTC write protection register (RTC_WPR)	939
29.7.10	RTC sub second register (RTC_SSR)	939
29.7.11	RTC shift control register (RTC_SHIFTR)	940
29.7.12	RTC timestamp time register (RTC_TSTR)	941
29.7.13	RTC timestamp date register (RTC_TSDDR)	942
29.7.14	RTC time-stamp sub second register (RTC_TSSSR)	943
29.7.15	RTC calibration register (RTC_CALR)	944
29.7.16	RTC tamper configuration register (RTC_TAMPCR)	945
29.7.17	RTC alarm A sub second register (RTC_ALRMASSR)	948
29.7.18	RTC alarm B sub second register (RTC_ALRMBSSR)	949
29.7.19	RTC option register (RTC_OR)	950
29.7.20	RTC backup registers (RTC_BKPxR)	950
29.7.21	RTC register map	951
<b>30</b>	<b>Independent watchdog (IWDG)</b>	<b>953</b>
30.1	Introduction	953
30.2	IWDG main features	953
30.3	IWDG functional description	953
30.3.1	IWDG block diagram	953
30.3.2	Window option	954
30.3.3	Hardware watchdog	955
30.3.4	Low-power freeze	955
30.3.5	Register access protection	955
30.3.6	Debug mode	955
30.4	IWDG registers	956
30.4.1	IWDG key register (IWDG_KR)	956
30.4.2	IWDG prescaler register (IWDG_PR)	957
30.4.3	IWDG reload register (IWDG_RLR)	958

---

30.4.4	IWDG status register (IWDG_SR) . . . . .	959
30.4.5	IWDG window register (IWDG_WINR) . . . . .	960
30.4.6	IWDG register map . . . . .	961
<b>31</b>	<b>System window watchdog (WWDG) . . . . .</b>	<b>962</b>
31.1	Introduction . . . . .	962
31.2	WWDG main features . . . . .	962
31.3	WWDG functional description . . . . .	962
31.3.1	WWDG block diagram . . . . .	963
31.3.2	Enabling the watchdog . . . . .	963
31.3.3	Controlling the down-counter . . . . .	963
31.3.4	How to program the watchdog timeout . . . . .	963
31.3.5	Debug mode . . . . .	965
31.4	WWDG interrupts . . . . .	965
31.5	WWDG registers . . . . .	965
31.5.1	WWDG control register (WWDG_CR) . . . . .	965
31.5.2	WWDG configuration register (WWDG_CFR) . . . . .	966
31.5.3	WWDG status register (WWDG_SR) . . . . .	967
31.5.4	WWDG register map . . . . .	967
<b>32</b>	<b>Inter-integrated circuit (I2C) interface . . . . .</b>	<b>968</b>
32.1	Introduction . . . . .	968
32.2	I2C main features . . . . .	968
32.3	I2C implementation . . . . .	969
32.4	I2C functional description . . . . .	969
32.4.1	I2C block diagram . . . . .	970
32.4.2	I2C pins and internal signals . . . . .	971
32.4.3	I2C clock requirements . . . . .	971
32.4.4	Mode selection . . . . .	971
32.4.5	I2C initialization . . . . .	972
32.4.6	Software reset . . . . .	977
32.4.7	Data transfer . . . . .	978
32.4.8	I2C slave mode . . . . .	980
32.4.9	I2C master mode . . . . .	989
32.4.10	I2C_TIMINGR register configuration examples . . . . .	1001
32.4.11	SMBus specific features . . . . .	1002

---

32.4.12	SMBus initialization . . . . .	1005
32.4.13	SMBus: I2C_TIMEOUTR register configuration examples . . . . .	1007
32.4.14	SMBus slave mode . . . . .	1007
32.4.15	Wake-up from Stop mode on address match . . . . .	1014
32.4.16	Error conditions . . . . .	1015
32.4.17	DMA requests . . . . .	1017
32.4.18	Debug mode . . . . .	1018
32.5	I2C low-power modes . . . . .	1018
32.6	I2C interrupts . . . . .	1019
32.7	I2C registers . . . . .	1020
32.7.1	I2C control register 1 (I2C_CR1) . . . . .	1020
32.7.2	I2C control register 2 (I2C_CR2) . . . . .	1023
32.7.3	I2C own address 1 register (I2C_OAR1) . . . . .	1025
32.7.4	I2C own address 2 register (I2C_OAR2) . . . . .	1026
32.7.5	I2C timing register (I2C_TIMINGR) . . . . .	1027
32.7.6	I2C timeout register (I2C_TIMEOUTR) . . . . .	1028
32.7.7	I2C interrupt and status register (I2C_ISR) . . . . .	1029
32.7.8	I2C interrupt clear register (I2C_ICR) . . . . .	1031
32.7.9	I2C PEC register (I2C_PECR) . . . . .	1032
32.7.10	I2C receive data register (I2C_RXDR) . . . . .	1033
32.7.11	I2C transmit data register (I2C_TXDR) . . . . .	1033
32.7.12	I2C register map . . . . .	1034
33	<b>Universal synchronous/asynchronous receiver transmitter (USART/UART) . . . . .</b>	<b>1036</b>
33.1	USART introduction . . . . .	1036
33.2	USART main features . . . . .	1037
33.3	USART extended features . . . . .	1038
33.4	USART implementation . . . . .	1038
33.5	USART functional description . . . . .	1039
33.5.1	USART block diagram . . . . .	1039
33.5.2	USART signals . . . . .	1040
33.5.3	USART character description . . . . .	1041
33.5.4	USART FIFOs and thresholds . . . . .	1043
33.5.5	USART transmitter . . . . .	1043
33.5.6	USART receiver . . . . .	1047

---

33.5.7	USART baud rate generation . . . . .	1054
33.5.8	Tolerance of the USART receiver to clock deviation . . . . .	1055
33.5.9	USART auto baud rate detection . . . . .	1057
33.5.10	USART multiprocessor communication . . . . .	1059
33.5.11	USART Modbus communication . . . . .	1061
33.5.12	USART parity control . . . . .	1062
33.5.13	USART LIN (local interconnection network) mode . . . . .	1063
33.5.14	USART synchronous mode . . . . .	1065
33.5.15	USART single-wire Half-duplex communication . . . . .	1069
33.5.16	USART receiver timeout . . . . .	1069
33.5.17	USART Smartcard mode . . . . .	1070
33.5.18	USART IrDA SIR ENDEC block . . . . .	1074
33.5.19	Continuous communication using USART and DMA . . . . .	1077
33.5.20	RS232 Hardware flow control and RS485 Driver Enable . . . . .	1079
33.5.21	USART low-power management . . . . .	1082
33.6	USART in low-power modes . . . . .	1085
33.7	USART interrupts . . . . .	1086
33.8	USART registers . . . . .	1087
33.8.1	USART control register 1 (USART_CR1) . . . . .	1087
33.8.2	USART control register 1 [alternate] (USART_CR1) . . . . .	1091
33.8.3	USART control register 2 (USART_CR2) . . . . .	1094
33.8.4	USART control register 3 (USART_CR3) . . . . .	1098
33.8.5	USART baud rate register (USART_BRR) . . . . .	1103
33.8.6	USART guard time and prescaler register (USART_GTPR) . . . . .	1103
33.8.7	USART receiver timeout register (USART_RTOR) . . . . .	1104
33.8.8	USART request register (USART_RQR) . . . . .	1105
33.8.9	USART interrupt and status register (USART_ISR) . . . . .	1106
33.8.10	USART interrupt and status register [alternate] (USART_ISR) . . . . .	1112
33.8.11	USART interrupt flag clear register (USART_ICR) . . . . .	1117
33.8.12	USART receive data register (USART_RDR) . . . . .	1119
33.8.13	USART transmit data register (USART_TDR) . . . . .	1119
33.8.14	USART prescaler register (USART_PRESC) . . . . .	1120
33.8.15	USART register map . . . . .	1121
<b>34</b>	<b>Low-power universal asynchronous receiver transmitter (LPUART) . . . . .</b>	<b>1123</b>
34.1	LPUART introduction . . . . .	1123

---

34.2	LPUART main features	1124
34.3	LPUART implementation	1125
34.4	LPUART functional description	1126
34.4.1	LPUART block diagram	1126
34.4.2	LPUART signals	1127
34.4.3	LPUART character description	1127
34.4.4	LPUART FIFOs and thresholds	1128
34.4.5	LPUART transmitter	1129
34.4.6	LPUART receiver	1132
34.4.7	LPUART baud rate generation	1136
34.4.8	Tolerance of the LPUART receiver to clock deviation	1137
34.4.9	LPUART multiprocessor communication	1138
34.4.10	LPUART parity control	1140
34.4.11	LPUART single-wire Half-duplex communication	1141
34.4.12	Continuous communication using DMA and LPUART	1141
34.4.13	RS232 Hardware flow control and RS485 Driver Enable	1144
34.4.14	LPUART low-power management	1146
34.5	LPUART in low-power modes	1149
34.6	LPUART interrupts	1150
34.7	LPUART registers	1151
34.7.1	LPUART control register 1 (LPUART_CR1)	1151
34.7.2	LPUART control register 1 [alternate] (LPUART_CR1)	1154
34.7.3	LPUART control register 2 (LPUART_CR2)	1157
34.7.4	LPUART control register 3 (LPUART_CR3)	1159
34.7.5	LPUART baud rate register (LPUART_BRR)	1162
34.7.6	LPUART request register (LPUART_RQR)	1163
34.7.7	LPUART interrupt and status register (LPUART_ISR)	1163
34.7.8	LPUART interrupt and status register [alternate] (LPUART_ISR)	1168
34.7.9	LPUART interrupt flag clear register (LPUART_ICR)	1171
34.7.10	LPUART receive data register (LPUART_RDR)	1172
34.7.11	LPUART transmit data register (LPUART_TDR)	1172
34.7.12	LPUART prescaler register (LPUART_PRESC)	1173
34.7.13	LPUART register map	1174
<b>35</b>	<b>Serial peripheral interface (SPI)</b>	<b>1176</b>
35.1	Introduction	1176

---

35.2	SPI main features	1176
35.3	SPI implementation	1176
35.4	SPI functional description	1177
35.4.1	General description	1177
35.4.2	Communications between one master and one slave	1178
35.4.3	Standard multi-slave communication	1180
35.4.4	Multi-master communication	1181
35.4.5	Slave select (NSS) pin management	1182
35.4.6	Communication formats	1183
35.4.7	Configuration of SPI	1185
35.4.8	Procedure for enabling SPI	1186
35.4.9	Data transmission and reception procedures	1186
35.4.10	SPI status flags	1196
35.4.11	SPI error flags	1197
35.4.12	NSS pulse mode	1198
35.4.13	TI mode	1198
35.4.14	CRC calculation	1199
35.5	SPI interrupts	1201
35.6	SPI registers	1202
35.6.1	SPI control register 1 (SPIx_CR1)	1202
35.6.2	SPI control register 2 (SPIx_CR2)	1204
35.6.3	SPI status register (SPIx_SR)	1206
35.6.4	SPI data register (SPIx_DR)	1207
35.6.5	SPI CRC polynomial register (SPIx_CRCPR)	1208
35.6.6	SPI Rx CRC register (SPIx_RXCRCR)	1208
35.6.7	SPI Tx CRC register (SPIx_TXCRCR)	1208
35.6.8	SPI register map	1210
<b>36</b>	<b>Serial audio interface (SAI)</b>	<b>1211</b>
36.1	Introduction	1211
36.2	SAI main features	1211
36.3	SAI implementation	1212
36.4	SAI functional description	1212
36.4.1	SAI block diagram	1212
36.4.2	SAI pins and internal signals	1214
36.4.3	Main SAI modes	1214

---

36.4.4	SAI synchronization mode . . . . .	1215
36.4.5	Audio data size . . . . .	1216
36.4.6	Frame synchronization . . . . .	1216
36.4.7	Slot configuration . . . . .	1219
36.4.8	SAI clock generator . . . . .	1221
36.4.9	Internal FIFOs . . . . .	1224
36.4.10	PDM interface . . . . .	1226
36.4.11	AC'97 link controller . . . . .	1234
36.4.12	SPDIF output . . . . .	1235
36.4.13	Specific features . . . . .	1238
36.4.14	Error flags . . . . .	1242
36.4.15	Disabling the SAI . . . . .	1245
36.4.16	SAI DMA interface . . . . .	1245
36.5	SAI interrupts . . . . .	1246
36.6	SAI registers . . . . .	1247
36.6.1	SAI configuration register 1 (SAI_ACR1) . . . . .	1247
36.6.2	SAI configuration register 1 (SAI_BCR1) . . . . .	1250
36.6.3	SAI configuration register 2 (SAI_ACR2) . . . . .	1252
36.6.4	SAI configuration register 2 (SAI_BCR2) . . . . .	1254
36.6.5	SAI frame configuration register (SAI_AFRCR) . . . . .	1256
36.6.6	SAI frame configuration register (SAI_BFRCR) . . . . .	1258
36.6.7	SAI slot register (SAI_ASLOTR) . . . . .	1259
36.6.8	SAI slot register (SAI_BSLOTR) . . . . .	1260
36.6.9	SAI interrupt mask register (SAI_AIM) . . . . .	1261
36.6.10	SAI interrupt mask register (SAI_BIM) . . . . .	1263
36.6.11	SAI status register (SAI_ASR) . . . . .	1264
36.6.12	SAI status register (SAI_BSR) . . . . .	1266
36.6.13	SAI clear flag register (SAI_ACLRFR) . . . . .	1268
36.6.14	SAI clear flag register (SAI_BCLRFR) . . . . .	1269
36.6.15	SAI data register (SAI_ADR) . . . . .	1270
36.6.16	SAI data register (SAI_BDR) . . . . .	1271
36.6.17	SAI PDM control register (SAI_PDMCR) . . . . .	1271
36.6.18	SAI PDM delay register (SAI_PDMDLY) . . . . .	1272
36.6.19	SAI register map . . . . .	1275
<b>37</b>	<b>Inter-processor communication controller (IPCC) . . . . .</b>	<b>1277</b>
37.1	IPCC introduction . . . . .	1277

---

37.2	IPCC main features .....	1277
37.3	IPCC functional description .....	1277
37.3.1	IPCC block diagram .....	1278
37.3.2	IPCC Simplex channel mode .....	1278
37.3.3	IPCC Half-duplex channel mode .....	1281
37.3.4	IPCC interrupts .....	1284
37.4	IPCC registers .....	1284
37.4.1	IPCC processor 1 control register (IPCC_C1CR) .....	1284
37.4.2	IPCC processor 1 mask register (IPCC_C1MR) .....	1285
37.4.3	IPCC processor 1 status set clear register (IPCC_C1SCR) .....	1286
37.4.4	IPCC processor 1 to processor 2 status register (IPCC_C1TOC2SR) .....	1286
37.4.5	IPCC processor 2 control register (IPCC_C2CR) .....	1287
37.4.6	IPCC processor 2 mask register (IPCC_C2MR) .....	1287
37.4.7	IPCC processor 2 status set clear register (IPCC_C2SCR) .....	1288
37.4.8	IPCC processor 2 to processor 1 status register (IPCC_C2TOC1SR) .....	1289
37.4.9	IPCC register map .....	1290
<b>38</b>	<b>Hardware semaphore (HSEM) .....</b>	<b>1291</b>
38.1	Introduction .....	1291
38.2	Main features .....	1291
38.3	Functional description .....	1292
38.3.1	HSEM block diagram .....	1292
38.3.2	HSEM internal signals .....	1292
38.3.3	HSEM lock procedures .....	1292
38.3.4	HSEM write/read/read lock register address .....	1294
38.3.5	HSEM unlock procedures .....	1294
38.3.6	HSEM COREID semaphore clear .....	1295
38.3.7	HSEM interrupts .....	1295
38.3.8	AHB bus master ID verification .....	1297
38.4	HSEM registers .....	1298
38.4.1	HSEM register semaphore x (HSEM_Rx) .....	1298
38.4.2	HSEM read lock register semaphore x (HSEM_RLRx) .....	1299
38.4.3	HSEM interrupt enable register (HSEM_CnIER) .....	1300
38.4.4	HSEM interrupt clear register (HSEM_CnICR) .....	1300
38.4.5	HSEM interrupt status register (HSEM_CnISR) .....	1300

---

38.4.6	HSEM interrupt status register (HSEM_CnMISR) . . . . .	1301
38.4.7	HSEM clear register (HSEM_CR) . . . . .	1301
38.4.8	HSEM clear semaphore key register (HSEM_KEYR) . . . . .	1302
38.4.9	HSEM register map . . . . .	1303
<b>39</b>	<b>Universal serial bus full-speed device interface (USB) . . . . .</b>	<b>1305</b>
39.1	Introduction . . . . .	1305
39.2	USB main features . . . . .	1305
39.3	USB implementation . . . . .	1305
39.4	USB functional description . . . . .	1306
39.4.1	Description of USB blocks . . . . .	1307
39.5	Programming considerations . . . . .	1308
39.5.1	Generic USB device programming . . . . .	1308
39.5.2	System and power-on reset . . . . .	1309
39.5.3	Double-buffered endpoints . . . . .	1314
39.5.4	Isochronous transfers . . . . .	1316
39.5.5	Suspend/Resume events . . . . .	1317
39.6	USB and USB SRAM registers . . . . .	1320
39.6.1	Common registers . . . . .	1320
39.6.2	Buffer descriptor table . . . . .	1333
39.6.3	USB register map . . . . .	1336
<b>40</b>	<b>Clock recovery system (CRS) . . . . .</b>	<b>1338</b>
40.1	Introduction . . . . .	1338
40.2	CRS main features . . . . .	1338
40.3	CRS implementation . . . . .	1338
40.4	CRS functional description . . . . .	1339
40.4.1	CRS block diagram . . . . .	1339
40.4.2	Synchronization input . . . . .	1339
40.4.3	Frequency error measurement . . . . .	1339
40.4.4	Frequency error evaluation and automatic trimming . . . . .	1340
40.4.5	CRS initialization and configuration . . . . .	1341
40.5	CRS low-power modes . . . . .	1342
40.6	CRS interrupts . . . . .	1342
40.7	CRS registers . . . . .	1343
40.7.1	CRS control register (CRS_CR) . . . . .	1343

---

40.7.2	CRS configuration register (CRS_CFGR) . . . . .	1344
40.7.3	CRS interrupt and status register (CRS_ISR) . . . . .	1345
40.7.4	CRS interrupt flag clear register (CRS_ICR) . . . . .	1347
40.7.5	CRS register map . . . . .	1347
<b>41</b>	<b>Debug support (DBG) . . . . .</b>	<b>1349</b>
41.1	Introduction . . . . .	1349
41.2	Debug use cases . . . . .	1349
41.3	DBG functional description . . . . .	1351
41.3.1	DBG block diagram . . . . .	1351
41.3.2	DBG pins and internal signals . . . . .	1351
41.3.3	DBG power domains . . . . .	1352
41.3.4	DBG clocks . . . . .	1352
41.3.5	Debug and low power modes . . . . .	1353
41.3.6	DBG reset . . . . .	1353
41.4	Serial wire and JTAG debug port (SWJ-DP) . . . . .	1353
41.4.1	JTAG debug port . . . . .	1353
41.4.2	SW debug port . . . . .	1356
41.4.3	Debug port registers . . . . .	1357
41.4.4	DP debug port identification register (DP_DPIDR) . . . . .	1358
41.4.5	DP abort register (DP_ABORTR) . . . . .	1358
41.4.6	DP control and status register (DP_CTRL/STATR) . . . . .	1359
41.4.7	DP data link control register (DP_DLCSR) . . . . .	1361
41.4.8	DP target identification register (DP_TARGETIDR) . . . . .	1361
41.4.9	DP data link protocol identification register (DP_DLPIIDR) . . . . .	1362
41.4.10	DP resend register (DP_RESENR) . . . . .	1362
41.4.11	DP access port select register (DP_SELECTR) . . . . .	1363
41.4.12	DP read buffer register (DP_BUFFR) . . . . .	1363
41.4.13	DP target selection register (DP_TARGETSELR) . . . . .	1364
41.4.14	Debug port register map and reset values . . . . .	1365
41.5	Access ports . . . . .	1366
41.5.1	AP control/status word register (AP_CSWR) . . . . .	1369
41.5.2	AP transfer address register (AP_TAR) . . . . .	1370
41.5.3	AP data read/write register (AP_DRWR) . . . . .	1370
41.5.4	AP banked data registers (AP_BD0-3R) . . . . .	1370
41.5.5	AP base address register (AP_BASER) . . . . .	1371
41.5.6	AP identification register (AP_IDR) . . . . .	1371

41.5.7	Access port register map and reset values	1373
41.6	Cross trigger interface (CTI) and matrix (CTM)	1374
41.7	Cross trigger interface registers	1378
41.7.1	CTI control register (CTI_CONTROLR)	1378
41.7.2	CTI trigger acknowledge register (CTI_INTACKR)	1378
41.7.3	CTI application trigger set register (CTI_APPSETR)	1378
41.7.4	CTI application trigger clear register (CTI_APPCLEAR)	1379
41.7.5	CTI application pulse register (CTI_APPPULSER)	1380
41.7.6	CTI trigger In x enable register (CTI_INENRx)	1380
41.7.7	CTI trigger out x enable register (CTI_OUTENRx)	1381
41.7.8	CTI trigger in status register (CTI_TRGISTSR)	1381
41.7.9	CTI trigger out status register (CTI_TRGOSTSR)	1382
41.7.10	CTI channel in status register (CTI_CHINSTSR)	1382
41.7.11	CTI channel out status register (CTI_CHOUTSTS)	1382
41.7.12	CTI channel gate register (CTI_GATER)	1383
41.7.13	CTI claim tag set register (CTI_CLAIMSETR)	1383
41.7.14	CTI claim tag clear register (CTI_CLAIMCLR)	1384
41.7.15	CTI lock access register (CTI_LAR)	1384
41.7.16	CTI lock status register (CTI_LSR)	1385
41.7.17	CTI authentication status register (CTI_AUTHSTATR)	1385
41.7.18	CTI device configuration register (CTI_DEVIDR)	1386
41.7.19	CTI device type identifier register (CTI_DEVTYPEP)	1386
41.7.20	CTI CoreSight peripheral identity register 4 (CTI_PIDR4)	1387
41.7.21	CTI CoreSight peripheral identity register 0 (CTI_PIDR0)	1387
41.7.22	CTI CoreSight peripheral identity register 1 (CTI_PIDR1)	1387
41.7.23	CTI CoreSight peripheral identity register 2 (CTI_PIDR2)	1388
41.7.24	CTI CoreSight peripheral identity register 3 (CTI_PIDR3)	1388
41.7.25	CTI CoreSight component identity register 0 (CTI_CIDR0)	1389
41.7.26	CTI CoreSight peripheral identity register 1 (CTI_CIDR1)	1389
41.7.27	CTI CoreSight component identity register 2 (CTI_CIDR2)	1390
41.7.28	CTI CoreSight component identity register 3 (CTI_CIDR3)	1390
41.7.29	CTI register map and reset values	1391
41.8	Microcontroller debug unit (DBGMCU)	1394
41.8.1	DBGMCU identity code register (DBGMCU_IDCODE)	1394
41.8.2	DBGMCU configuration register (DBGMCU_CR)	1394
41.8.3	DBGMCU CPU1 APB1 peripheral freeze register 1 (DBGMCU_APB1FZR1)	1395

---

41.8.4	DBGMCU CPU2 APB1 peripheral freeze register 1 (DBGMCU_C2APB1FZR1) . . . . .	1396
41.8.5	DBGMCU CPU1 APB1 peripheral freeze register 2 (DBGMCU_APB1FZR2) . . . . .	1397
41.8.6	DBGMCU CPU2 APB1 peripheral freeze register 2 (DBGMCU_C2APB1FZR2) . . . . .	1398
41.8.7	DBGMCU CPU1 APB2 peripheral freeze register (DBGMCU_APB2FZR) . . . . .	1398
41.8.8	DBGMCU CPU2 APB2 peripheral freeze register (DBGMCU_C2APB2FZR) . . . . .	1399
41.8.9	DBGMCU register map and reset values . . . . .	1401
41.9	CPU2 ROM tables . . . . .	1403
41.9.1	CPU2 ROM1 memory type register (C2ROM1_MEMTYPER) . . . . .	1405
41.9.2	CPU2 ROM1 CoreSight peripheral identity register 4 (C2ROM1_PIDR4) . . . . .	1405
41.9.3	CPU2 ROM1 CoreSight peripheral identity register 0 (C2ROM1_PIDR0) . . . . .	1405
41.9.4	CPU2 ROM1 CoreSight peripheral identity register 1 (C2ROM1_PIDR1) . . . . .	1406
41.9.5	CPU2 ROM1 CoreSight peripheral identity register 2 (C2ROM1_PIDR2) . . . . .	1406
41.9.6	CPU2 ROM1 CoreSight peripheral identity register 3 (C2ROM1_PIDR3) . . . . .	1407
41.9.7	CPU2 ROM1 CoreSight component identity register 0 (C2ROM1_CIDR0) . . . . .	1407
41.9.8	CPU2 ROM1 CoreSight peripheral identity register 1 (C2ROM1_CIDR1) . . . . .	1408
41.9.9	CPU2 ROM1 CoreSight component identity register 2 (C2ROM1_CIDR2) . . . . .	1408
41.9.10	CPU2 ROM1 CoreSight component identity register 3 (C2ROM1_CIDR3) . . . . .	1408
41.9.11	CPU2 processor ROM table registers and reset values . . . . .	1410
41.9.12	CPU2 ROM2 memory type register (C2ROM2_MEMTYPER) . . . . .	1411
41.9.13	CPU2 ROM2 CoreSight peripheral identity register 4 (C2ROM2_PIDR4) . . . . .	1411
41.9.14	CPU2 ROM2 CoreSight peripheral identity register 0 (C2ROM2_PIDR0) . . . . .	1411
41.9.15	CPU2 ROM2 CoreSight peripheral identity register 1 (C2ROM2_PIDR1) . . . . .	1412
41.9.16	CPU2 ROM2 CoreSight peripheral identity register 2 (C2ROM2_PIDR2) . . . . .	1412

41.9.17	CPU2 ROM2 CoreSight peripheral identity register 3 (C2ROM2_PIDR3) .....	1413
41.9.18	CPU2 ROM2 CoreSight component identity register 0 (C2ROM2_CIDR0) .....	1413
41.9.19	CPU2 ROM2 CoreSight peripheral identity register 1 (C2ROM2_CIDR1) .....	1414
41.9.20	CPU2 ROM2 CoreSight component identity register 2 (C2ROM2_CIDR2) .....	1414
41.9.21	CPU2 ROM2 CoreSight component identity register 3 (C2ROM2_CIDR3) .....	1414
41.9.22	CPU2 ROM table register map and reset values .....	1416
41.10	CPU2 data watchpoint and trace unit (DWT) .....	1417
41.10.1	DWT control register (DWT_CTRLR) .....	1417
41.10.2	DWT cycle count register (DWT_CYCCNTR) .....	1419
41.10.3	DWT CPI count register (DWT_CPICNTR) .....	1419
41.10.4	DWT exception count register (DWT_EXCCNTR) .....	1420
41.10.5	DWT sleep count register (DWT_SLPCNTR) .....	1420
41.10.6	DWT LSU count register (DWT_LSUCNTR) .....	1421
41.10.7	DWT fold count register (DWT_FOLDCNTR) .....	1421
41.10.8	DWT program counter sample register (DWT_PCSR) .....	1421
41.10.9	DWT comparator register x (DWT_COMPxR) .....	1422
41.10.10	DWT mask register x (DWT_MASKxR) .....	1422
41.10.11	DWT function register x (DWT_FUNCTxR) .....	1422
41.10.12	DWT CoreSight peripheral identity register 4 (DWT_PIDR4) .....	1423
41.10.13	DWT CoreSight peripheral identity register 0 (DWT_PIDR0) .....	1424
41.10.14	DWT CoreSight peripheral identity register 1 (DWT_PIDR1) .....	1424
41.10.15	DWT CoreSight peripheral identity register 2 (DWT_PIDR2) .....	1425
41.10.16	DWT CoreSight peripheral identity register 3 (DWT_PIDR3) .....	1425
41.10.17	DWT CoreSight component identity register 0 (DWT_CIDR0) .....	1426
41.10.18	DWT CoreSight peripheral identity register 1 (DWT_CIDR1) .....	1426
41.10.19	DWT CoreSight component identity register 2 (DWT_CIDR2) .....	1426
41.10.20	DWT CoreSight component identity register 3 (DWT_CIDR3) .....	1427
41.10.21	CPU2 DWT registers .....	1428
41.11	CPU2 breakpoint unit (PBU) .....	1431
41.11.1	BPU control register (BPU_CTRLR) .....	1431
41.11.2	BPU remap register (BPU_REMAPR) .....	1431
41.11.3	BPU comparator registers (BPU_COMPxR) .....	1432
41.11.4	BPU CoreSight peripheral identity register 4 (BPU_PIDR4) .....	1432

41.11.5	BPU CoreSight peripheral identity register 0 (BPU_PIDR0) . . . . .	1433
41.11.6	BPU CoreSight peripheral identity register 1 (BPU_PIDR1) . . . . .	1433
41.11.7	BPU CoreSight peripheral identity register 2 (BPU_PIDR2) . . . . .	1433
41.11.8	BPU CoreSight peripheral identity register 3 (BPU_PIDR3) . . . . .	1434
41.11.9	BPU CoreSight component identity register 0 (BPU_CIDR0) . . . . .	1434
41.11.10	BPU CoreSight peripheral identity register 1 (BPU_CIDR1) . . . . .	1435
41.11.11	BPU CoreSight component identity register 2 (BPU_CIDR2) . . . . .	1435
41.11.12	BPU CoreSight component identity register 3 (BPU_CIDR3) . . . . .	1436
41.11.13	CPU2 BPU register map and reset values . . . . .	1437
41.12	CPU2 cross trigger interface (CTI) . . . . .	1438
41.13	CPU1 ROM table . . . . .	1438
41.13.1	CPU1 ROM memory type register (C1ROM_MEMTYPER) . . . . .	1439
41.13.2	CPU1 ROM CoreSight peripheral identity register 4 (C1ROM_PIDR4) . . . . .	1440
41.13.3	CPU1 ROM CoreSight peripheral identity register 0 (C1ROM_PIDR0) . . . . .	1440
41.13.4	CPU1 ROM CoreSight peripheral identity register 1 (C1ROM_PIDR1) . . . . .	1441
41.13.5	CPU1 ROM CoreSight peripheral identity register 2 (C1ROM_PIDR2) . . . . .	1441
41.13.6	CPU1 ROM CoreSight peripheral identity register 3 (C1ROM_PIDR3) . . . . .	1442
41.13.7	CPU1 ROM CoreSight component identity register 0 (C1ROM_CIDR0) . . . . .	1442
41.13.8	CPU1 ROM CoreSight peripheral identity register 1 (C1ROM_CIDR1) . . . . .	1442
41.13.9	CPU1 ROM CoreSight component identity register 2 (C1ROM_CIDR2) . . . . .	1443
41.13.10	CPU1 ROM CoreSight component identity register 3 (C1ROM_CIDR3) . . . . .	1443
41.13.11	CPU1 ROM table register map and reset values . . . . .	1445
41.14	CPU1 data watchpoint and trace unit (DWT) . . . . .	1446
41.14.1	DWT control register (DWT_CTRLR) . . . . .	1446
41.14.2	DWT cycle count register (DWT_CYCCNTR) . . . . .	1448
41.14.3	DWT CPI count register (DWT_CPICNTR) . . . . .	1448
41.14.4	DWT exception count register (DWT_EXCCNTR) . . . . .	1449
41.14.5	DWT sleep count register (DWT_SLPCNTR) . . . . .	1449
41.14.6	DWT LSU count register (DWT_LSUCNTR) . . . . .	1450
41.14.7	DWT fold count register (DWT_FOLDCNTR) . . . . .	1450

41.14.8	DWT program counter sample register (DWT_PCSR) . . . . .	1450
41.14.9	DWT comparator register x (DWT_COMPxR) . . . . .	1451
41.14.10	DWT mask register x (DWT_MASKxR) . . . . .	1451
41.14.11	DWT function register x (DWT_FUNCTxR) . . . . .	1451
41.14.12	DWT CoreSight peripheral identity register 4 (DWT_PIDR4) . . . . .	1452
41.14.13	DWT CoreSight peripheral identity register 0 (DWT_PIDR0) . . . . .	1453
41.14.14	DWT CoreSight peripheral identity register 1 (DWT_PIDR1) . . . . .	1453
41.14.15	DWT CoreSight peripheral identity register 2 (DWT_PIDR2) . . . . .	1454
41.14.16	DWT CoreSight peripheral identity register 3 (DWT_PIDR3) . . . . .	1454
41.14.17	DWT CoreSight component identity register 0 (DWT_CIDR0) . . . . .	1455
41.14.18	DWT CoreSight peripheral identity register 1 (DWT_CIDR1) . . . . .	1455
41.14.19	DWT CoreSight component identity register 2 (DWT_CIDR2) . . . . .	1455
41.14.20	DWT CoreSight component identity register 3 (DWT_CIDR3) . . . . .	1456
41.14.21	CPU1 DWT register map and reset values . . . . .	1457
41.15	CPU1 instrumentation trace macrocell (ITM) . . . . .	1459
41.15.1	ITM stimulus register x (ITM_STIMRx) . . . . .	1459
41.15.2	ITM trace enable register (ITM_TER) . . . . .	1459
41.15.3	ITM trace privilege register (ITM_TPR) . . . . .	1460
41.15.4	ITM trace control register (ITM_TCR) . . . . .	1460
41.15.5	ITM CoreSight peripheral identity register 4 (ITM_PIDR4) . . . . .	1461
41.15.6	ITM CoreSight peripheral identity register 0 (ITM_PIDR0) . . . . .	1462
41.15.7	ITM CoreSight peripheral identity register 1 (ITM_PIDR1) . . . . .	1462
41.15.8	ITM CoreSight peripheral identity register 2 (ITM_PIDR2) . . . . .	1463
41.15.9	ITM CoreSight peripheral identity register 3 (ITM_PIDR3) . . . . .	1463
41.15.10	ITM CoreSight component identity register 0 (ITM_CIDR0) . . . . .	1464
41.15.11	ITM CoreSight peripheral identity register 1 (ITM_CIDR1) . . . . .	1464
41.15.12	ITM CoreSight component identity register 2 (ITM_CIDR2) . . . . .	1464
41.15.13	ITM CoreSight component identity register 3 (ITM_CIDR3) . . . . .	1465
41.15.14	ITM register map and reset values . . . . .	1466
41.16	CPU1 breakpoint unit (FPB) . . . . .	1467
41.16.1	FPB control register (FPB_CTRLR) . . . . .	1467
41.16.2	FPB remap register (FPB_REMAPR) . . . . .	1467
41.16.3	FPB comparator registers (FPB_COMPxR) . . . . .	1468
41.16.4	FPB CoreSight peripheral identity register 4 (FPB_PIDR4) . . . . .	1468
41.16.5	FPB CoreSight peripheral identity register 0 (FPB_PIDR0) . . . . .	1469
41.16.6	FPB CoreSight peripheral identity register 1 (FPB_PIDR1) . . . . .	1469
41.16.7	FPB CoreSight peripheral identity register 2 (FPB_PIDR2) . . . . .	1470

41.16.8	FPB CoreSight peripheral identity register 3 (FPB_PIDR3) . . . . .	1470
41.16.9	FPB CoreSight component identity register 0 (FPB_CIDR0) . . . . .	1471
41.16.10	FPB CoreSight peripheral identity register 1 (FPB_CIDR1) . . . . .	1471
41.16.11	FPB CoreSight component identity register 2 (FPB_CIDR2) . . . . .	1471
41.16.12	FPB CoreSight component identity register 3 (FPB_CIDR3) . . . . .	1472
41.16.13	FPB register map and reset values . . . . .	1473
41.17	CPU1 Embedded trace macrocell (ETM™), only available on STM32WB55xx . . . . .	1474
41.17.1	ETM control register (ETM_CR) . . . . .	1474
41.17.2	ETM configuration code register (ETM_CCR) . . . . .	1475
41.17.3	ETM trigger register (ETM_TRIGGER) . . . . .	1476
41.17.4	ETM status register (ETM_SR) . . . . .	1477
41.17.5	ETM status register (ETM_SCR) . . . . .	1477
41.17.6	ETM trace enable event register (ETM_TEEVR) . . . . .	1478
41.17.7	ETM trace enable control 1 register (ETM_TECR1) . . . . .	1479
41.17.8	ETM FIFOFULL level register (ETM_FFLR) . . . . .	1479
41.17.9	ETM counter reload value 1 register (ETM_CNTRLDVR1) . . . . .	1480
41.17.10	ETM synchronization frequency register (ETM_SYNCFR) . . . . .	1480
41.17.11	ETM ID register (ETM_IDR) . . . . .	1481
41.17.12	ETM configuration code extension register (ETM_CCER) . . . . .	1481
41.17.13	ETM trace enable start/stop EmbeddedICE control register (ETM_TESSEICR) . . . . .	1482
41.17.14	ETM timestamp event register (ETM_TSEVR) . . . . .	1483
41.17.15	ETM trace ID register (ETM_TRACEIDR) . . . . .	1484
41.17.16	ETM ID register 2(ETM_IDR2) . . . . .	1484
41.17.17	ETM device power down status register 2(ETM_PDSR) . . . . .	1484
41.17.18	ETM claim tag set register (ETM_CLAIMSETR) . . . . .	1485
41.17.19	ETM claim tag clear register (ETM_CLAIMCLR) . . . . .	1485
41.17.20	ETM lock access register (ETM_LAR) . . . . .	1486
41.17.21	ETM lock status register (ETM_LSR) . . . . .	1486
41.17.22	ETM authentication status register (ETM_AUTHSTATR) . . . . .	1487
41.17.23	ETM CoreSight device identity register (ETM_DEVTYPE) . . . . .	1487
41.17.24	ETM CoreSight peripheral identity register 4 (ETM_PIDR4) . . . . .	1488
41.17.25	ETM CoreSight peripheral identity register 0 (ETM_PIDR0) . . . . .	1488
41.17.26	ETM CoreSight peripheral identity register 1 (ETM_PIDR1) . . . . .	1488
41.17.27	ETM CoreSight peripheral identity register 2 (ETM_PIDR2) . . . . .	1489
41.17.28	ETM CoreSight peripheral identity register 3 (ETM_PIDR3) . . . . .	1489

---

41.17.29	ETM CoreSight component identity register 0 (ETM_CIDR0) . . . . .	1490
41.17.30	ETM CoreSight peripheral identity register 1 (ETM_CIDR1) . . . . .	1490
41.17.31	ETM CoreSight component identity register 2 (ETM_CIDR2) . . . . .	1491
41.17.32	ETM CoreSight component identity register 3 (ETM_CIDR3) . . . . .	1491
41.17.33	ETM register map and reset values . . . . .	1492
41.18	CPU1 trace port interface unit (TPIU) . . . . .	1495
41.18.1	TPIU supported port size register (TPIU_SSPSR) . . . . .	1495
41.18.2	TPIU current port size register (TPIU_CSPPSR) . . . . .	1496
41.18.3	TPIU asynchronous clock prescaler register (TPIU_ACPR) . . . . .	1496
41.18.4	TPIU selected pin protocol register (TPIU_SPPR) . . . . .	1496
41.18.5	TPIU formatter and flush status register (TPIU_FFSR) . . . . .	1497
41.18.6	TPIU formatter and flush control register (TPIU_FFCR) . . . . .	1497
41.18.7	TPIU formatter synchronization counter register (TPIU_FSCR) . . . . .	1498
41.18.8	TPIU claim tag set register (TPIU_CLAIMSETR) . . . . .	1498
41.18.9	TPIU claim tag clear register (TPIU_CLAIMCLR) . . . . .	1499
41.18.10	TPIU device configuration register (TPIU_DEVIDR) . . . . .	1499
41.18.11	TPIU device type identifier register (TPIU_DEVTYPE) . . . . .	1500
41.18.12	TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4) . . . . .	1500
41.18.13	TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0) . . . . .	1501
41.18.14	TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1) . . . . .	1501
41.18.15	TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2) . . . . .	1502
41.18.16	TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3) . . . . .	1502
41.18.17	TPIU CoreSight component identity register 0 (TPIU_CIDR0) . . . . .	1503
41.18.18	TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1) . . . . .	1503
41.18.19	TPIU CoreSight component identity register 2 (TPIU_CIDR2) . . . . .	1503
41.18.20	TPIU CoreSight component identity register 3 (TPIU_CIDR3) . . . . .	1504
41.18.21	CPU1 TPIU register map and reset values . . . . .	1505
41.19	CPU1 cross trigger interface (CTI) . . . . .	1507
41.20	References . . . . .	1507
<b>42</b>	<b>Device electronic signature . . . . .</b>	<b>1508</b>
42.1	Unique device ID register (96 bits) . . . . .	1508
42.2	Memory size data register . . . . .	1509
42.2.1	Flash size data register . . . . .	1509
42.3	Package data register . . . . .	1509
42.4	Part number codification register . . . . .	1510

<b>43</b>	<b>Important security notice</b>	<b>1511</b>
<b>44</b>	<b>Revision history</b>	<b>1512</b>

## List of tables

Table 1.	Memory map and peripheral register boundary addresses . . . . .	68
Table 2.	Boot modes. . . . .	73
Table 3.	Flash memory - Single bank organization . . . . .	76
Table 4.	Number of wait states vs, Flash memory clock (HCLK4) frequency. . . . .	78
Table 5.	Page erase overview . . . . .	83
Table 6.	Mass erase overview . . . . .	84
Table 7.	Errors in page-based row programming . . . . .	89
Table 8.	Option bytes format . . . . .	90
Table 9.	Option bytes organization . . . . .	90
Table 10.	Option loading control . . . . .	99
Table 11.	UID64 organization . . . . .	100
Table 12.	Flash memory read protection status . . . . .	101
Table 13.	RDP regression from Level 1 to Level 0 and memory erase . . . . .	103
Table 14.	Access status vs. protection level and execution modes . . . . .	104
Table 17.	Flash memory interrupt requests . . . . .	109
Table 18.	Flash interface register map and reset values . . . . .	128
Table 19.	CRC internal input/output signals . . . . .	133
Table 20.	CRC register map and reset values . . . . .	138
Table 21.	Supply configuration control . . . . .	142
Table 22.	PVM features . . . . .	148
Table 23.	Sub-system low power wakeup sources. . . . .	152
Table 24.	Low-power mode summary . . . . .	154
Table 25.	Functionalities depending on system operating mode . . . . .	155
Table 26.	Low-power run . . . . .	158
Table 27.	CPU CSTOP wakeup vs. system operating mode . . . . .	159
Table 28.	Sleep mode. . . . .	160
Table 29.	Low-power sleep. . . . .	161
Table 30.	Stop0 mode . . . . .	163
Table 31.	Stop1 mode . . . . .	164
Table 32.	Stop2 mode . . . . .	167
Table 33.	Standby mode. . . . .	168
Table 34.	Shutdown mode . . . . .	170
Table 35.	PWR register map and reset values . . . . .	192
Table 36.	Interconnect matrix implementation . . . . .	194
Table 37.	Peripherals interconnect matrix . . . . .	194
Table 38.	Maximum clock source frequency . . . . .	212
Table 39.	SMPS step-down converter clock source selection and division . . . . .	214
Table 40.	Peripheral clock enable . . . . .	218
Table 41.	Single core Low power debug configurations . . . . .	219
Table 42.	RCC register map and reset values . . . . .	283
Table 43.	GPIO implementation . . . . .	289
Table 44.	Port bit configuration table . . . . .	291
Table 45.	GPIO register map and reset values . . . . .	306
Table 46.	SYSCFG register map and reset values. . . . .	323
Table 47.	DMA1 and DMA2 implementation . . . . .	326
Table 48.	DMA internal input/output signals . . . . .	328
Table 49.	Programmable data width and endian behavior (when PINC = MINC = 1) . . . . .	333
Table 50.	DMA interrupt requests . . . . .	335

Table 51.	DMA register map and reset values	343
Table 52.	DMAMUX implementation	347
Table 53.	DMAMUX instantiation	347
Table 54.	DMAMUX: assignment of multiplexer inputs to resources	348
Table 55.	DMAMUX: assignment of trigger inputs to resources	348
Table 56.	DMAMUX: assignment of synchronization inputs to resources	349
Table 57.	DMAMUX signals	351
Table 58.	DMAMUX interrupts	355
Table 59.	DMAMUX register map and reset values	360
Table 60.	NVIC implementation	362
Table 61.	CPU1 vector table	364
Table 62.	CPU2 vector table	367
Table 63.	Wakeup interrupt table	369
Table 64.	EXTI implementation	371
Table 65.	EXTI pin overview	372
Table 66.	EVG pin overview	373
Table 67.	EXTI event input configurations and register control	374
Table 68.	Masking functionality	377
Table 69.	EXTI register map sections	378
Table 70.	EXTI register map and reset values	388
Table 71.	QUADSPI pins	391
Table 72.	QUADSPI interrupt requests	403
Table 73.	QUADSPI register map and reset values	414
Table 74.	ADC features	417
Table 75.	ADC internal input/output signals	419
Table 76.	ADC input/output pins	419
Table 77.	Configuring the trigger polarity for regular external triggers	435
Table 78.	Configuring the trigger polarity for injected external triggers	435
Table 79.	ADC1 - External triggers for regular channels	436
Table 80.	ADC1 - External trigger for injected channels	436
Table 81.	TSAR timings depending on resolution	448
Table 82.	Offset computation versus data resolution	451
Table 83.	Analog watchdog channel selection	461
Table 84.	Analog watchdog 1 comparison	462
Table 85.	Analog watchdog 2 and 3 comparison	462
Table 86.	Maximum output results versus N and M (gray cells indicate truncation)	466
Table 87.	Effect of low-power modes on the ADC	474
Table 88.	ADC interrupts	475
Table 89.	ADC register map and reset values	504
Table 90.	ADC register map and reset values (master and slave ADC common registers) offset = 0x300	506
Table 91.	VREFBUF implementation	507
Table 92.	VREF buffer modes	507
Table 93.	VREFBUF trimming data	508
Table 94.	VREFBUF register map and reset values	509
Table 95.	COMP1 input plus assignment	512
Table 96.	COMP1 input minus assignment	513
Table 97.	COMP2 input plus assignment	513
Table 98.	COMP2 input minus assignment	513
Table 99.	Comparator behavior in the low power modes	517
Table 100.	Interrupt control bits	517
Table 101.	COMP register map and reset values	523

---

Table 102. Example of frame rate calculation . . . . .	527
Table 103. Blink frequency . . . . .	535
Table 104. Remapping capability . . . . .	539
Table 105. LCD interrupt requests . . . . .	546
Table 106. LCD register map and reset values . . . . .	553
Table 107. Acquisition sequence summary . . . . .	558
Table 108. Spread spectrum deviation versus AHB clock frequency . . . . .	560
Table 109. I/O state depending on its mode and IODEF bit value . . . . .	561
Table 110. Effect of low-power modes on TSC . . . . .	563
Table 111. Interrupt control bits . . . . .	563
Table 112. TSC register map and reset values . . . . .	572
Table 113. RNG internal input/output signals . . . . .	575
Table 114. RNG interrupt requests . . . . .	582
Table 115. RNG register map and reset map . . . . .	585
Table 116. AES internal input/output signals . . . . .	587
Table 117. CTR mode initialization vector definition . . . . .	604
Table 118. GCM last block definition . . . . .	606
Table 119. Initialization of AES_IVRx registers in GCM mode . . . . .	607
Table 120. Initialization of AES_IVRx registers in CCM mode . . . . .	614
Table 121. Key endianness in AES_KEYRx registers (128- or 256-bit key length) . . . . .	619
Table 122. AES interrupt requests . . . . .	622
Table 123. Processing latency for ECB, CBC and CTR . . . . .	622
Table 124. Processing latency for GCM and CCM (in clock cycles) . . . . .	623
Table 125. AES register map and reset values . . . . .	633
Table 126. Internal input/output signals . . . . .	636
Table 127. PKA integer arithmetic functions list . . . . .	637
Table 128. PKA prime field ( $F_p$ ) elliptic curve functions list . . . . .	637
Table 129. Montgomery parameter computation . . . . .	642
Table 130. Modular addition . . . . .	643
Table 131. Modular subtraction . . . . .	643
Table 132. Montgomery multiplication . . . . .	644
Table 133. Modular exponentiation (normal mode) . . . . .	645
Table 134. Modular exponentiation (fast mode) . . . . .	645
Table 135. Modular inversion . . . . .	645
Table 136. Modular reduction . . . . .	646
Table 137. Arithmetic addition . . . . .	646
Table 138. Arithmetic subtraction . . . . .	646
Table 139. Arithmetic multiplication . . . . .	647
Table 140. Arithmetic comparison . . . . .	647
Table 141. CRT exponentiation . . . . .	648
Table 142. Point on elliptic curve $F_p$ check . . . . .	649
Table 143. ECC $F_p$ scalar multiplication . . . . .	649
Table 144. ECC $F_p$ scalar multiplication (Fast Mode) . . . . .	650
Table 145. ECDSA sign - Inputs . . . . .	651
Table 146. ECDSA sign - Outputs . . . . .	651
Table 147. Extended ECDSA sign (extra outputs) . . . . .	652
Table 148. ECDSA verification (inputs) . . . . .	652
Table 149. ECDSA verification (outputs) . . . . .	652
Table 150. Family of supported curves for ECC operations . . . . .	653
Table 151. Modular exponentiation computation times . . . . .	655
Table 152. ECC scalar multiplication computation times . . . . .	655
Table 153. ECDSA signature average computation times . . . . .	655

Table 154. ECDSA verification average computation times . . . . .	656
Table 155. Point on elliptic curve Fp check average computation times . . . . .	656
Table 156. Montgomery parameters average computation times . . . . .	656
Table 157. PKA interrupt requests . . . . .	656
Table 158. PKA register map and reset values . . . . .	660
Table 159. Behavior of timer outputs versus BRK/BRK2 inputs . . . . .	703
Table 160. Break protection disarming conditions . . . . .	705
Table 161. Counting direction versus encoder signals . . . . .	711
Table 162. TIM1 internal trigger connection . . . . .	728
Table 163. Output control bits for complementary OCx and OCxN channels with break feature . . . . .	742
Table 164. TIM1 register map and reset values . . . . .	759
Table 165. Counting direction versus encoder signals . . . . .	795
Table 166. TIM2 internal trigger connection . . . . .	812
Table 167. Output control bit for standard OCx channels . . . . .	823
Table 168. TIM2 register map and reset values . . . . .	830
Table 169. Break protection disarming conditions . . . . .	855
Table 170. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) . . . . .	872
Table 171. TIM16/TIM17 register map and reset values . . . . .	883
Table 172. LPTIM features . . . . .	886
Table 173. LPTIM implementation . . . . .	886
Table 174. LPTIM1 external trigger connection . . . . .	887
Table 175. LPTIM2 external trigger connection . . . . .	888
Table 176. Prescaler division ratios . . . . .	889
Table 177. Encoder counting scenarios . . . . .	896
Table 178. Effect of low-power modes on the LPTIM . . . . .	897
Table 179. Interrupt events . . . . .	898
Table 180. LPTIM register map and reset values . . . . .	909
Table 181. RTC implementation . . . . .	912
Table 182. Effect of low-power modes on RTC . . . . .	926
Table 183. Interrupt control bits . . . . .	926
Table 184. RTC register map and reset values . . . . .	951
Table 185. IWDG register map and reset values . . . . .	961
Table 186. WWDG register map and reset values . . . . .	967
Table 187. I2C implementation . . . . .	969
Table 188. I2C input/output pins . . . . .	971
Table 189. I2C internal input/output signals . . . . .	971
Table 190. Comparison of analog vs. digital filters . . . . .	973
Table 191. I2C-SMBus specification data setup and hold times . . . . .	976
Table 192. I2C configuration . . . . .	980
Table 193. I2C-SMBus specification clock timings . . . . .	991
Table 194. Examples of timing settings for fI2CCLK = 8 MHz . . . . .	1001
Table 195. Examples of timing settings for fI2CCLK = 16 MHz . . . . .	1001
Table 196. SMBus timeout specifications . . . . .	1004
Table 197. SMBus with PEC configuration . . . . .	1005
Table 198. Examples of TIMEOUTA settings for various I2CCLK frequencies (max $t_{TIMEOUT} = 25$ ms) . . . . .	1007
Table 199. Examples of TIMEOUTB settings for various I2CCLK frequencies . . . . .	1007
Table 200. Examples of TIMEOUTA settings for various I2CCLK frequencies (max $t_{IDLE} = 50$ $\mu$ s) . . . . .	1007
Table 201. Effect of low-power modes on the I2C . . . . .	1018
Table 202. I2C Interrupt requests . . . . .	1019

---

Table 203. I2C register map and reset values . . . . .	1034
Table 204. USART / LPUART features . . . . .	1038
Table 205. Noise detection from sampled data . . . . .	1053
Table 206. Tolerance of the USART receiver when BRR [3:0] = 0000. . . . .	1056
Table 207. Tolerance of the USART receiver when BRR[3:0] is different from 0000. . . . .	1057
Table 208. USART frame formats . . . . .	1062
Table 209. Effect of low-power modes on the USART . . . . .	1085
Table 210. USART interrupt requests. . . . .	1086
Table 211. USART register map and reset values. . . . .	1121
Table 212. USART / LPUART features . . . . .	1125
Table 213. Error calculation for programmed baud rates at lpuart_ker_ck_pres = 32.768 kHz . . . . .	1136
Table 214. Error calculation for programmed baud rates at fCK = 100 MHz . . . . .	1137
Table 215. Tolerance of the LPUART receiver. . . . .	1138
Table 217. Effect of low-power modes on the LPUART . . . . .	1149
Table 218. LPUART interrupt requests. . . . .	1150
Table 219. LPUART register map and reset values. . . . .	1174
Table 220. SPI implementation. . . . .	1177
Table 221. SPI interrupt requests . . . . .	1201
Table 222. SPI register map and reset values . . . . .	1210
Table 223. STM32WB55xx SAI features . . . . .	1212
Table 224. SAI internal input/output signals . . . . .	1214
Table 225. SAI input/output pins. . . . .	1214
Table 226. MCLK_x activation conditions. . . . .	1221
Table 227. Clock generator programming examples . . . . .	1224
Table 228. TDM settings. . . . .	1231
Table 229. TDM frame configuration examples . . . . .	1233
Table 230. SOPD pattern . . . . .	1236
Table 231. Parity bit calculation . . . . .	1236
Table 232. Audio sampling frequency versus symbol rates . . . . .	1237
Table 233. SAI interrupt sources . . . . .	1246
Table 234. SAI register map and reset values . . . . .	1275
Table 235. IPCC interface signals . . . . .	1278
Table 236. Bits used for the communication. . . . .	1279
Table 237. IPCC register map and reset values. . . . .	1290
Table 238. HSEM internal input/output signals. . . . .	1292
Table 239. Authorized AHB bus master IDs . . . . .	1297
Table 240. HSEM register map and reset values. . . . .	1303
Table 241. USB implementation. . . . .	1305
Table 242. Double-buffering buffer flag definition. . . . .	1315
Table 243. Bulk double-buffering memory buffers usage. . . . .	1315
Table 244. Isochronous memory buffers usage . . . . .	1317
Table 245. Resume event detection. . . . .	1318
Table 246. Reception status encoding . . . . .	1331
Table 247. Endpoint type encoding . . . . .	1331
Table 248. Endpoint kind meaning . . . . .	1331
Table 249. Transmission status encoding . . . . .	1332
Table 250. Definition of allocated buffer memory . . . . .	1335
Table 251. USB register map and reset values . . . . .	1336
Table 252. CRS features . . . . .	1338
Table 253. Effect of low-power modes on CRS . . . . .	1342
Table 254. Interrupt control bits . . . . .	1342
Table 255. CRS register map and reset values . . . . .	1347

---

Table 256. JTAG/Serial-wire debug port pins . . . . .	1351
Table 257. Trace port pins . . . . .	1352
Table 258. Single Wire Trace port pins . . . . .	1352
Table 259. Trigger pins . . . . .	1352
Table 260. JTAG-DP data registers . . . . .	1355
Table 261. Packet request . . . . .	1356
Table 262. ACK response . . . . .	1357
Table 263. Data transfer . . . . .	1357
Table 264. Debug port register map and reset values . . . . .	1365
Table 265. Access port register map and reset values . . . . .	1373
Table 266. CPU2 CTI inputs . . . . .	1374
Table 267. CPU2 CTI outputs . . . . .	1375
Table 268. CPU1 CTI inputs . . . . .	1375
Table 269. CPU1 CTI outputs . . . . .	1375
Table 270. CTI register map and reset values . . . . .	1391
Table 271. DBGMCU register map and reset values . . . . .	1401
Table 272. CPU2 processor ROM table . . . . .	1403
Table 273. CPU2 ROM table . . . . .	1403
Table 274. CPU2 processor ROM table register map and reset values . . . . .	1410
Table 275. CPU2 ROM table register map and reset values . . . . .	1416
Table 276. CPU2 DWT register map and reset values . . . . .	1428
Table 277. CPU2 BPU register map and reset values . . . . .	1437
Table 278. CPU1 ROM table . . . . .	1438
Table 279. CPU1 ROM table register map and reset values . . . . .	1445
Table 280. CPU1 DWT register map and reset values . . . . .	1457
Table 281. CPU1 ITM register map and reset values . . . . .	1466
Table 282. CPU1 FPB register map and reset values . . . . .	1473
Table 283. CPU1 ETM register map and reset values . . . . .	1492
Table 284. CPU1 TPIU register map and reset values . . . . .	1505
Table 285. Document revision history . . . . .	1512

# List of figures

Figure 1.	System architecture . . . . .	64
Figure 2.	Memory map . . . . .	67
Figure 3.	Sequential 16-bit instructions execution . . . . .	80
Figure 4.	Changing the Read protection (RDP) level . . . . .	104
Figure 5.	Radio system block diagram . . . . .	131
Figure 6.	CRC calculation unit block diagram . . . . .	133
Figure 7.	Power supply overview . . . . .	141
Figure 8.	Supply configurations . . . . .	141
Figure 9.	Brown-out reset waveform . . . . .	147
Figure 10.	PVD thresholds . . . . .	148
Figure 11.	CPU2 boot options . . . . .	150
Figure 12.	Low-power modes possible transitions . . . . .	153
Figure 13.	Real-time radio activity flags . . . . .	171
Figure 14.	Simplified diagram of the reset circuit . . . . .	201
Figure 15.	Clock tree . . . . .	205
Figure 16.	HSE clock sources . . . . .	206
Figure 17.	LSE clock sources . . . . .	210
Figure 18.	Frequency measurement with TIM16 in capture mode . . . . .	216
Figure 19.	Frequency measurement with TIM17 in capture mode . . . . .	216
Figure 20.	Three-volt or five-volt tolerant GPIO structure (TT or FT) . . . . .	290
Figure 21.	Input floating/pull up/pull down configurations . . . . .	295
Figure 22.	Output configuration . . . . .	295
Figure 23.	Alternate function configuration . . . . .	296
Figure 24.	High impedance-analog configuration . . . . .	297
Figure 25.	DMA block diagram . . . . .	327
Figure 26.	DMAMUX block diagram . . . . .	350
Figure 27.	Synchronization mode of the DMAMUX request line multiplexer channel . . . . .	353
Figure 28.	Event generation of the DMA request line multiplexer channel . . . . .	353
Figure 29.	Interrupt block diagram . . . . .	363
Figure 30.	EXTI block diagram . . . . .	372
Figure 31.	Configurable event trigger logic CPU wakeup . . . . .	375
Figure 32.	Direct event trigger logic CPU wakeup . . . . .	376
Figure 33.	QUADSPI block diagram . . . . .	390
Figure 34.	Example of read command in quad-SPI mode . . . . .	391
Figure 35.	Example of a DDR command in quad-SPI mode . . . . .	395
Figure 36.	NCS when CKMODE = 0 (T = CLK period) . . . . .	402
Figure 37.	NCS when CKMODE = 1 in SDR mode (T = CLK period) . . . . .	402
Figure 38.	NCS when CKMODE = 1 in DDR mode (T = CLK period) . . . . .	402
Figure 39.	NCS when CKMODE = 1 with an abort (T = CLK period) . . . . .	403
Figure 40.	ADC block diagram . . . . .	418
Figure 41.	ADC clock scheme . . . . .	421
Figure 42.	ADC1 connectivity . . . . .	422
Figure 43.	ADC calibration . . . . .	425
Figure 44.	Updating the ADC calibration factor . . . . .	426
Figure 45.	Mixing single-ended and differential channels . . . . .	427
Figure 46.	Enabling / disabling the ADC . . . . .	428
Figure 47.	Analog to digital conversion time . . . . .	433
Figure 48.	Stopping ongoing regular conversions . . . . .	434

Figure 49. Stopping ongoing regular and injected conversions . . . . .	434
Figure 50. Injected conversion latency . . . . .	438
Figure 51. Example of JSQR queue of context (sequence change) . . . . .	441
Figure 52. Example of JSQR queue of context (trigger change) . . . . .	441
Figure 53. Example of JSQR queue of context with overflow before conversion . . . . .	442
Figure 54. Example of JSQR queue of context with overflow during conversion . . . . .	442
Figure 55. Example of JSQR queue of context with empty queue (case JQM = 0) . . . . .	443
Figure 56. Example of JSQR queue of context with empty queue (case JQM = 1) . . . . .	444
Figure 57. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs during an ongoing conversion . . . . .	444
Figure 58. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs during an ongoing conversion and a new trigger occurs . . . . .	445
Figure 59. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs outside an ongoing conversion . . . . .	445
Figure 60. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1) . . . . .	446
Figure 61. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0) . . . . .	446
Figure 62. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 1) . . . . .	447
Figure 63. Single conversions of a sequence, software trigger . . . . .	449
Figure 64. Continuous conversion of a sequence, software trigger . . . . .	449
Figure 65. Single conversions of a sequence, hardware trigger . . . . .	450
Figure 66. Continuous conversions of a sequence, hardware trigger . . . . .	450
Figure 67. Right alignment (offset disabled, unsigned value) . . . . .	452
Figure 68. Right alignment (offset enabled, signed value) . . . . .	453
Figure 69. Left alignment (offset disabled, unsigned value) . . . . .	453
Figure 70. Left alignment (offset enabled, signed value) . . . . .	454
Figure 71. Example of overrun (OVR) . . . . .	455
Figure 72. AUTODLY = 1, regular conversion in continuous mode, software trigger . . . . .	458
Figure 73. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) . . . . .	458
Figure 74. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1) . . . . .	459
Figure 75. AUTODLY = 1, regular continuous conversions interrupted by injected conversions . . . . .	460
Figure 76. AUTODLY = 1 in auto- injected mode (JAUTO = 1) . . . . .	460
Figure 77. Analog watchdog guarded area . . . . .	461
Figure 78. ADCy_AWDx_OUT signal generation (on all regular channels) . . . . .	463
Figure 79. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software) . . . . .	464
Figure 80. ADCy_AWDx_OUT signal generation (on a single regular channel) . . . . .	464
Figure 81. ADCy_AWDx_OUT signal generation (on all injected channels) . . . . .	464
Figure 82. 20-bit to 16-bit result truncation . . . . .	465
Figure 83. Numerical example with 5-bit shift and rounding . . . . .	465
Figure 84. Triggered regular oversampling mode (TROVS bit = 1) . . . . .	467
Figure 85. Regular oversampling modes (4x ratio) . . . . .	468
Figure 86. Regular and injected oversampling modes used simultaneously . . . . .	469
Figure 87. Triggered regular oversampling with injection . . . . .	469
Figure 88. Oversampling in auto-injected mode . . . . .	470
Figure 89. Temperature sensor channel block diagram . . . . .	471
Figure 90. VBAT channel block diagram . . . . .	472
Figure 91. VREFINT channel block diagram . . . . .	473
Figure 92. Comparator block diagram . . . . .	512
Figure 93. Window mode . . . . .	515
Figure 94. Comparator hysteresis . . . . .	515

---

Figure 95. Comparator output blanking .....	516
Figure 96. LCD controller block diagram .....	526
Figure 97. 1/3 bias, 1/4 duty .....	528
Figure 98. Static duty case 1 .....	529
Figure 99. Static duty case 2 .....	530
Figure 100. 1/2 duty, 1/2 bias .....	531
Figure 101. 1/3 duty, 1/3 bias .....	532
Figure 102. 1/4 duty, 1/3 bias .....	533
Figure 103. 1/8 duty, 1/4 bias .....	534
Figure 104. LCD voltage control .....	537
Figure 105. Deadtime .....	538
Figure 106. SEG/COM mux feature example .....	544
Figure 107. Flowchart example .....	545
Figure 108. TSC block diagram .....	556
Figure 109. Surface charge transfer analog I/O group structure .....	557
Figure 110. Sampling capacitor voltage variation .....	558
Figure 111. Charge transfer acquisition sequence .....	559
Figure 112. Spread spectrum variation principle .....	560
Figure 113. RNG block diagram .....	575
Figure 114. Entropy source model .....	576
Figure 115. RNG initialization overview .....	579
Figure 116. AES block diagram .....	587
Figure 117. ECB encryption and decryption principle .....	589
Figure 118. CBC encryption and decryption principle .....	590
Figure 119. CTR encryption and decryption principle .....	591
Figure 120. GCM encryption and authentication principle .....	592
Figure 121. GMAC authentication principle .....	592
Figure 122. CCM encryption and authentication principle .....	593
Figure 123. Encryption key derivation for ECB/CBC decryption (Mode 2) .....	596
Figure 124. Example of suspend mode management .....	597
Figure 125. ECB encryption .....	598
Figure 126. ECB decryption .....	598
Figure 127. CBC encryption .....	599
Figure 128. CBC decryption .....	599
Figure 129. ECB/CBC encryption (Mode 1) .....	600
Figure 130. ECB/CBC decryption (Mode 3) .....	601
Figure 131. Message construction in CTR mode .....	603
Figure 132. CTR encryption .....	604
Figure 133. CTR decryption .....	604
Figure 134. Message construction in GCM .....	606
Figure 135. GCM authenticated encryption .....	607
Figure 136. Message construction in GMAC mode .....	611
Figure 137. GMAC authentication mode .....	611
Figure 138. Message construction in CCM mode .....	612
Figure 139. CCM mode authenticated encryption .....	614
Figure 140. 128-bit block construction with respect to data swap .....	618
Figure 141. DMA transfer of a 128-bit data block during input phase .....	620
Figure 142. DMA transfer of a 128-bit data block during output phase .....	621
Figure 143. PKA block diagram .....	636
Figure 144. Advanced-control timer block diagram .....	663
Figure 145. Counter timing diagram with prescaler division change from 1 to 2 .....	665
Figure 146. Counter timing diagram with prescaler division change from 1 to 4 .....	665

---

Figure 147. Counter timing diagram, internal clock divided by 1 .....	667
Figure 148. Counter timing diagram, internal clock divided by 2 .....	667
Figure 149. Counter timing diagram, internal clock divided by 4 .....	668
Figure 150. Counter timing diagram, internal clock divided by N .....	668
Figure 151. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) .....	669
Figure 152. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) .....	669
Figure 153. Counter timing diagram, internal clock divided by 1 .....	671
Figure 154. Counter timing diagram, internal clock divided by 2 .....	671
Figure 155. Counter timing diagram, internal clock divided by 4 .....	672
Figure 156. Counter timing diagram, internal clock divided by N .....	672
Figure 157. Counter timing diagram, update event when repetition counter is not used .....	673
Figure 158. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 .....	674
Figure 159. Counter timing diagram, internal clock divided by 2 .....	675
Figure 160. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 .....	675
Figure 161. Counter timing diagram, internal clock divided by N .....	676
Figure 162. Counter timing diagram, update event with ARPE=1 (counter underflow) .....	676
Figure 163. Counter timing diagram, Update event with ARPE=1 (counter overflow) .....	677
Figure 164. Update rate examples depending on mode and TIMx_RCR register settings .....	678
Figure 165. External trigger input block .....	679
Figure 166. TIM1 ETR input circuitry .....	679
Figure 167. Control circuit in normal mode, internal clock divided by 1 .....	680
Figure 168. TI2 external clock connection example .....	681
Figure 169. Control circuit in external clock mode 1 .....	682
Figure 170. External trigger input block .....	682
Figure 171. Control circuit in external clock mode 2 .....	683
Figure 172. Capture/compare channel (example: channel 1 input stage) .....	684
Figure 173. Capture/compare channel 1 main circuit .....	684
Figure 174. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3) .....	685
Figure 175. Output stage of capture/compare channel (channel 4) .....	685
Figure 176. Output stage of capture/compare channel (channel 5, idem ch. 6) .....	686
Figure 177. PWM input mode timing .....	688
Figure 178. Output compare mode, toggle on OC1 .....	690
Figure 179. Edge-aligned PWM waveforms (ARR=8) .....	691
Figure 180. Center-aligned PWM waveforms (ARR=8) .....	692
Figure 181. Generation of 2 phase-shifted PWM signals with 50% duty cycle .....	694
Figure 182. Combined PWM mode on channel 1 and 3 .....	695
Figure 183. 3-phase combined PWM signals with multiple trigger pulses per period .....	696
Figure 184. Complementary output with dead-time insertion .....	697
Figure 185. Dead-time waveforms with delay greater than the negative pulse .....	697
Figure 186. Dead-time waveforms with delay greater than the positive pulse .....	698
Figure 187. Break and Break2 circuitry overview .....	700
Figure 188. Various output behavior in response to a break event on BRK (OSSI = 1) .....	702
Figure 189. PWM output state following BRK and BRK2 pins assertion (OSSI=1) .....	703
Figure 190. PWM output state following BRK assertion (OSSI=0) .....	704
Figure 191. Output redirection (BRK2 request not represented) .....	705
Figure 192. Clearing TIMx OCxREF .....	706
Figure 193. 6-step generation, COM example (OSSR=1) .....	707
Figure 194. Example of one pulse mode .....	708
Figure 195. Retriggerable one pulse mode .....	710
Figure 196. Example of counter operation in encoder interface mode .....	711
Figure 197. Example of encoder interface mode with TI1FP1 polarity inverted .....	712
Figure 198. Measuring time interval between edges on 3 signals .....	713

---

Figure 199. Example of Hall sensor interface . . . . .	715
Figure 200. Control circuit in reset mode . . . . .	716
Figure 201. Control circuit in Gated mode . . . . .	717
Figure 202. Control circuit in trigger mode . . . . .	718
Figure 203. Control circuit in external clock mode 2 + trigger mode . . . . .	719
Figure 204. General-purpose timer block diagram . . . . .	763
Figure 205. Counter timing diagram with prescaler division change from 1 to 2 . . . . .	765
Figure 206. Counter timing diagram with prescaler division change from 1 to 4 . . . . .	765
Figure 207. Counter timing diagram, internal clock divided by 1 . . . . .	766
Figure 208. Counter timing diagram, internal clock divided by 2 . . . . .	767
Figure 209. Counter timing diagram, internal clock divided by 4 . . . . .	767
Figure 210. Counter timing diagram, internal clock divided by N . . . . .	768
Figure 211. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded) . . . . .	768
Figure 212. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded) . . . . .	769
Figure 213. Counter timing diagram, internal clock divided by 1 . . . . .	770
Figure 214. Counter timing diagram, internal clock divided by 2 . . . . .	770
Figure 215. Counter timing diagram, internal clock divided by 4 . . . . .	771
Figure 216. Counter timing diagram, internal clock divided by N . . . . .	771
Figure 217. Counter timing diagram, Update event when repetition counter is not used . . . . .	772
Figure 218. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 . . . . .	773
Figure 219. Counter timing diagram, internal clock divided by 2 . . . . .	774
Figure 220. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . . .	774
Figure 221. Counter timing diagram, internal clock divided by N . . . . .	775
Figure 222. Counter timing diagram, Update event with ARPE=1 (counter underflow) . . . . .	775
Figure 223. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . . .	776
Figure 224. Control circuit in normal mode, internal clock divided by 1 . . . . .	777
Figure 225. TI2 external clock connection example . . . . .	777
Figure 226. Control circuit in external clock mode 1 . . . . .	778
Figure 227. External trigger input block . . . . .	779
Figure 228. Control circuit in external clock mode 2 . . . . .	780
Figure 229. Capture/Compare channel (example: channel 1 input stage) . . . . .	780
Figure 230. Capture/Compare channel 1 main circuit . . . . .	781
Figure 231. Output stage of Capture/Compare channel (channel 1) . . . . .	781
Figure 232. PWM input mode timing . . . . .	783
Figure 233. Output compare mode, toggle on OC1 . . . . .	785
Figure 234. Edge-aligned PWM waveforms (ARR=8) . . . . .	786
Figure 235. Center-aligned PWM waveforms (ARR=8) . . . . .	788
Figure 236. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . . .	789
Figure 237. Combined PWM mode on channels 1 and 3 . . . . .	790
Figure 238. Clearing TIMx OCxREF . . . . .	791
Figure 239. Example of one-pulse mode . . . . .	792
Figure 240. Retriggerable one-pulse mode . . . . .	794
Figure 241. Example of counter operation in encoder interface mode . . . . .	795
Figure 242. Example of encoder interface mode with TI1FP1 polarity inverted . . . . .	796
Figure 243. Control circuit in reset mode . . . . .	797
Figure 244. Control circuit in gated mode . . . . .	798
Figure 245. Control circuit in trigger mode . . . . .	799
Figure 246. Control circuit in external clock mode 2 + trigger mode . . . . .	800
Figure 247. Master/Slave timer example . . . . .	800
Figure 248. Master/slave connection example with 1 channel only timers . . . . .	801
Figure 249. Gating TIM2 with OC1REF of TIM1 . . . . .	802

---

Figure 250. Gating TIM2 with Enable of TIM1 . . . . .	803
Figure 251. Triggering TIM2 with update of TIM1 . . . . .	803
Figure 252. Triggering TIM2 with Enable of TIM1 . . . . .	804
Figure 253. TIM16/TIM17 block diagram . . . . .	834
Figure 254. Counter timing diagram with prescaler division change from 1 to 2 . . . . .	836
Figure 255. Counter timing diagram with prescaler division change from 1 to 4 . . . . .	836
Figure 256. Counter timing diagram, internal clock divided by 1 . . . . .	838
Figure 257. Counter timing diagram, internal clock divided by 2 . . . . .	838
Figure 258. Counter timing diagram, internal clock divided by 4 . . . . .	839
Figure 259. Counter timing diagram, internal clock divided by N . . . . .	839
Figure 260. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) . . . . .	840
Figure 261. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . . .	840
Figure 262. Update rate examples depending on mode and TIMx_RCR register settings . . . . .	842
Figure 263. Control circuit in normal mode, internal clock divided by 1 . . . . .	843
Figure 264. TI2 external clock connection example . . . . .	843
Figure 265. Control circuit in external clock mode 1 . . . . .	844
Figure 266. Capture/compare channel (example: channel 1 input stage) . . . . .	845
Figure 267. Capture/compare channel 1 main circuit . . . . .	845
Figure 268. Output stage of capture/compare channel (channel 1) . . . . .	846
Figure 269. Output compare mode, toggle on OC1 . . . . .	849
Figure 270. Edge-aligned PWM waveforms (ARR=8) . . . . .	850
Figure 271. Complementary output with dead-time insertion . . . . .	851
Figure 272. Dead-time waveforms with delay greater than the negative pulse . . . . .	851
Figure 273. Dead-time waveforms with delay greater than the positive pulse . . . . .	852
Figure 274. Output behavior in response to a break . . . . .	854
Figure 275. Output redirection . . . . .	856
Figure 276. 6-step generation, COM example (OSSR=1) . . . . .	857
Figure 277. Example of one pulse mode . . . . .	858
Figure 278. Low-power timer block diagram . . . . .	887
Figure 279. Glitch filter timing diagram . . . . .	889
Figure 280. LPTIM output waveform, single counting mode configuration . . . . .	891
Figure 281. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set) . . . . .	891
Figure 282. LPTIM output waveform, Continuous counting mode configuration . . . . .	892
Figure 283. Waveform generation . . . . .	893
Figure 284. Encoder mode counting sequence . . . . .	897
Figure 285. IRTIM internal hardware connections with TIM16 and TIM17 . . . . .	910
Figure 286. RTC block diagram . . . . .	913
Figure 287. Independent watchdog block diagram . . . . .	953
Figure 288. Watchdog block diagram . . . . .	963
Figure 289. Window watchdog timing diagram . . . . .	964
Figure 290. I2C block diagram . . . . .	970
Figure 291. I2C bus protocol . . . . .	972
Figure 292. Setup and hold timings . . . . .	974
Figure 293. I2C initialization flow . . . . .	977
Figure 294. Data reception . . . . .	978
Figure 295. Data transmission . . . . .	979
Figure 296. Slave initialization flow . . . . .	982
Figure 297. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0 . . . . .	984
Figure 298. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1 . . . . .	985

Figure 299. Transfer bus diagrams for I2C slave transmitter (mandatory events only) . . . . .	986
Figure 300. Transfer sequence flow for slave receiver with NOSTRETCH = 0 . . . . .	987
Figure 301. Transfer sequence flow for slave receiver with NOSTRETCH = 1 . . . . .	988
Figure 302. Transfer bus diagrams for I2C slave receiver (mandatory events only) . . . . .	988
Figure 303. Master clock generation . . . . .	990
Figure 304. Master initialization flow . . . . .	992
Figure 305. 10-bit address read access with HEAD10R = 0 . . . . .	992
Figure 306. 10-bit address read access with HEAD10R = 1 . . . . .	993
Figure 307. Transfer sequence flow for I2C master transmitter for $N \leq 255$ bytes . . . . .	994
Figure 308. Transfer sequence flow for I2C master transmitter for $N > 255$ bytes . . . . .	995
Figure 309. Transfer bus diagrams for I2C master transmitter (mandatory events only) . . . . .	996
Figure 310. Transfer sequence flow for I2C master receiver for $N \leq 255$ bytes . . . . .	998
Figure 311. Transfer sequence flow for I2C master receiver for $N > 255$ bytes . . . . .	999
Figure 312. Transfer bus diagrams for I2C master receiver (mandatory events only) . . . . .	1000
Figure 313. Timeout intervals for $t_{LOW:SEXT}$ , $t_{LOW:MEXT}$ . . . . .	1004
Figure 314. Transfer sequence flow for SMBus slave transmitter $N$ bytes + PEC . . . . .	1008
Figure 315. Transfer bus diagrams for SMBus slave transmitter (SBC = 1) . . . . .	1008
Figure 316. Transfer sequence flow for SMBus slave receiver $N$ bytes + PEC . . . . .	1010
Figure 317. Bus transfer diagrams for SMBus slave receiver (SBC = 1) . . . . .	1011
Figure 318. Bus transfer diagrams for SMBus master transmitter . . . . .	1012
Figure 319. Bus transfer diagrams for SMBus master receiver . . . . .	1014
Figure 320. USART block diagram . . . . .	1039
Figure 321. Word length programming . . . . .	1042
Figure 322. Configurable stop bits . . . . .	1044
Figure 323. TC/TXE behavior when transmitting . . . . .	1047
Figure 324. Start bit detection when oversampling by 16 or 8 . . . . .	1048
Figure 325. usart_ker_ck clock divider block diagram . . . . .	1051
Figure 326. Data sampling when oversampling by 16 . . . . .	1052
Figure 327. Data sampling when oversampling by 8 . . . . .	1053
Figure 328. Mute mode using Idle line detection . . . . .	1060
Figure 329. Mute mode using address mark detection . . . . .	1061
Figure 330. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . . .	1064
Figure 331. Break detection in LIN mode vs. Framing error detection . . . . .	1065
Figure 332. USART example of synchronous master transmission . . . . .	1066
Figure 333. USART data clock timing diagram in synchronous master mode (M bits = 00) . . . . .	1066
Figure 334. USART data clock timing diagram in synchronous master mode (M bits = 01) . . . . .	1067
Figure 335. USART data clock timing diagram in synchronous slave mode (M bits = 00) . . . . .	1068
Figure 336. ISO 7816-3 asynchronous protocol . . . . .	1070
Figure 337. Parity error detection using the 1.5 stop bits . . . . .	1072
Figure 338. IrDA SIR ENDEC block diagram . . . . .	1076
Figure 339. IrDA data modulation (3/16) - Normal mode . . . . .	1076
Figure 340. Transmission using DMA . . . . .	1078
Figure 341. Reception using DMA . . . . .	1079
Figure 342. Hardware flow control between 2 USARTs . . . . .	1079
Figure 343. RS232 RTS flow control . . . . .	1080
Figure 344. RS232 CTS flow control . . . . .	1081

Figure 345. Wakeup event verified (wakeup event = address match, FIFO disabled) . . . . .	1084
Figure 346. Wakeup event not verified (wakeup event = address match, FIFO disabled) . . . . .	1084
Figure 347. LPUART block diagram . . . . .	1126
Figure 348. LPUART word length programming . . . . .	1128
Figure 349. Configurable stop bits . . . . .	1130
Figure 350. TC/TXE behavior when transmitting . . . . .	1132
Figure 351. Ipuart_ker_ck clock divider block diagram . . . . .	1135
Figure 352. Mute mode using Idle line detection . . . . .	1139
Figure 353. Mute mode using address mark detection . . . . .	1140
Figure 354. Transmission using DMA . . . . .	1142
Figure 355. Reception using DMA . . . . .	1143
Figure 356. Hardware flow control between 2 LPUARTs . . . . .	1144
Figure 357. RS232 RTS flow control . . . . .	1144
Figure 358. RS232 CTS flow control . . . . .	1145
Figure 359. Wakeup event verified (wakeup event = address match, FIFO disabled) . . . . .	1148
Figure 360. Wakeup event not verified (wakeup event = address match, FIFO disabled) . . . . .	1148
Figure 361. SPI block diagram . . . . .	1177
Figure 362. Full-duplex single master/ single slave application . . . . .	1178
Figure 363. Half-duplex single master/ single slave application . . . . .	1179
Figure 364. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode) . . . . .	1180
Figure 365. Master and three independent slaves . . . . .	1181
Figure 366. Multi-master application . . . . .	1182
Figure 367. Hardware/software slave select management . . . . .	1183
Figure 368. Data clock timing diagram . . . . .	1184
Figure 369. Data alignment when data length is not equal to 8-bit or 16-bit . . . . .	1185
Figure 370. Packing data in FIFO for transmission and reception . . . . .	1189
Figure 371. Master full-duplex communication . . . . .	1192
Figure 372. Slave full-duplex communication . . . . .	1193
Figure 373. Master full-duplex communication with CRC . . . . .	1194
Figure 374. Master full-duplex communication in packed mode . . . . .	1195
Figure 375. NSSP pulse generation in Motorola SPI master mode . . . . .	1198
Figure 376. TI mode transfer . . . . .	1199
Figure 377. SAI functional block diagram . . . . .	1213
Figure 378. Audio frame . . . . .	1216
Figure 379. FS role is start of frame + channel side identification (FSDEF = TRIS = 1) . . . . .	1218
Figure 380. FS role is start of frame (FSDEF = 0) . . . . .	1219
Figure 381. Slot size configuration with FBOFF = 0 in SAI_xSLOTR . . . . .	1220
Figure 382. First bit offset . . . . .	1220
Figure 383. Audio block clock generator overview . . . . .	1222
Figure 384. PDM typical connection and timing . . . . .	1226
Figure 385. Detailed PDM interface block diagram . . . . .	1227
Figure 386. Start-up sequence . . . . .	1228
Figure 387. SAI_ADR format in TDM, 32-bit slot width . . . . .	1229
Figure 388. SAI_ADR format in TDM, 16-bit slot width . . . . .	1230
Figure 389. SAI_ADR format in TDM, 8-bit slot width . . . . .	1231
Figure 390. AC'97 audio frame . . . . .	1234
Figure 391. SPDIF format . . . . .	1235
Figure 392. SAI_xDR register ordering . . . . .	1236

---

Figure 393. Data companding hardware in an audio block in the SAI . . . . .	1240
Figure 394. Tristate strategy on SD output line on an inactive slot . . . . .	1241
Figure 395. Tristate on output data line in a protocol like I2S . . . . .	1242
Figure 396. Overrun detection error . . . . .	1243
Figure 397. FIFO underrun event . . . . .	1243
Figure 398. IPCC block diagram . . . . .	1278
Figure 399. IPCC Simplex channel mode transfer timing . . . . .	1279
Figure 400. IPCC Simplex - Send procedure state diagram . . . . .	1280
Figure 401. IPCC Simplex - Receive procedure state diagram . . . . .	1281
Figure 402. IPCC Half-duplex channel mode transfer timing . . . . .	1282
Figure 403. IPCC Half-duplex - Send procedure state diagram . . . . .	1282
Figure 404. IPCC Half-duplex - Receive procedure state diagram . . . . .	1283
Figure 405. HSEM block diagram . . . . .	1292
Figure 406. Procedure state diagram . . . . .	1293
Figure 407. Interrupt state diagram . . . . .	1296
Figure 408. USB peripheral block diagram . . . . .	1306
Figure 409. Packet buffer areas with examples of buffer description table locations . . . . .	1310
Figure 410. CRS block diagram . . . . .	1339
Figure 411. CRS counter behavior . . . . .	1340
Figure 412. Block diagram of debug support infrastructure . . . . .	1351
Figure 413. JTAG TAP state machine . . . . .	1354
Figure 414. Debug and access port connections . . . . .	1366
Figure 415. Debugger connection to debug components . . . . .	1368
Figure 416. Embedded cross trigger . . . . .	1374
Figure 417. Mapping trigger inputs to outputs . . . . .	1376
Figure 418. Cross trigger configuration example . . . . .	1377
Figure 419. CPU2 CoreSight™ topology . . . . .	1404
Figure 420. CPU1 CoreSight™ topology . . . . .	1439
Figure 421. Trace port interface unit (TPIU) . . . . .	1495

# 1 Documentation conventions

## 1.1 General information

The STM32WB55xx/STM32WB35xx devices embed an Arm<sup>®(a)</sup> CPU1 Cortex<sup>®</sup>-M4 core.



## 1.2 List of abbreviations for registers

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

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

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

## 1.3 Glossary

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

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **Option bytes**: product configuration bits stored in the flash memory.
- **AHB**: advanced high-performance bus.

## 1.4 Availability of peripherals

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

## 2 System and memory overview

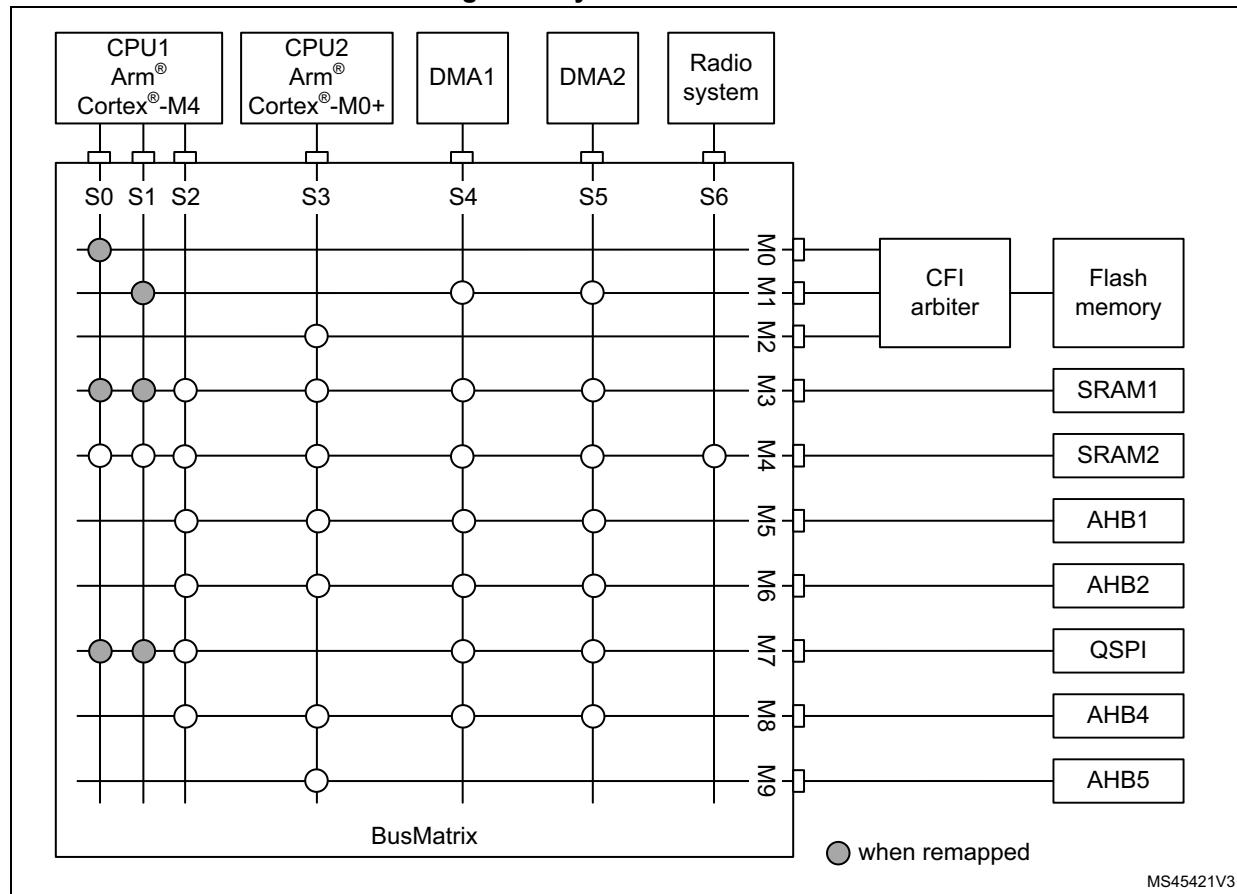
### 2.1 System architecture

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

- Seven masters:
  - CPU1 (CPU1 Cortex®-M4 with FPU) core I-bus
  - CPU1 (CPU1 Cortex®-M4 with FPU) core D-bus
  - CPU1 (CPU1 Cortex®-M4 with FPU) core S-bus
  - CPU2 (Cortex® -M0+) core S-bus
  - DMA1
  - DMA2
  - Radio system
- Ten slaves:
  - Internal Flash memory on the CPU1 (CPU1 Cortex®-M4) ICode bus
  - Internal Flash memory on CPU1 (CPU1 Cortex®-M4) DCode bus
  - Internal Flash memory on CPU2 (Cortex® -M0+) S bus
  - Internal SRAM1 (up to 192 KB)
  - Internal SRAM2a (32 KB) + SRAM2b (32 KB)
  - AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - Quad SPI memory interface (QUADSPI)
  - AHB4 shared peripheral
  - AHB5 including AHB to APB bridge and Radio peripherals (connected to APB3)

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](#).

Figure 1. System architecture



### 2.1.1 S0: CPU1 (CPU1 Cortex®-M4) I-bus

This bus connects the instruction bus of the CPU1 core to the BusMatrix. This bus is used by the core to fetch instructions. The targets of this bus are the internal Flash memory, SRAM1, SRAM2a (backup), SRAM2b (non backup) and the external memories through QUADSPI.

### 2.1.2 S1: CPU1 (CPU1 Cortex®-M4) D-bus

This bus connects the data bus of the CPU1 core to the BusMatrix. This bus is used by the core for literal load and debug access. The targets of this bus are the internal Flash memory, SRAM1, SRAM2a (backup), SRAM2b (non backup) and the external memories through QUADSPI.

### 2.1.3 S2: CPU1 (CPU1 Cortex®-M4) S-bus

This bus connects the system bus of the CPU1 core to the BusMatrix. This bus is used by the core to access data located in a peripheral or SRAM area. The targets of this bus are SRAM1, SRAM2a (backup), SRAM2b (non backup), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, the AHB4 peripherals and the external memories through QUADSPI.

## 2.1.4 S3: CPU2 (Cortex® -M0+) S-bus

This bus connects the system bus of the CPU2 core to the BusMatrix. This bus is used by the core to fetch instructions, for literal load and debug access, and access data located in a peripheral or SRAM area. The targets of this bus are the internal Flash memory, SRAM1, SRAM2a (backup), SRAM2b (non backup), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, and the AHB4 peripherals and the AHB5 peripherals including the APB3 peripherals.

## 2.1.5 S4, S5: DMA-bus

This bus connects the AHB master interface of the DMA to the BusMatrix. The targets of this bus are the SRAM1, SRAM2a (backup), SRAM2b (non backup), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, the AHB4 peripherals and the external memories through QUADSPI.

## 2.1.6 S6: Radio system-bus

This bus connects the AHB master interface of the Radio system to the BusMatrix. The targets of this bus are the SRAM2a (backup) and SRAM2b (non backup).

## 2.1.7 BusMatrix

The BusMatrix manages the access arbitration between masters. The arbitration uses a Round Robin algorithm. The BusMatrix is composed by seven masters (CPU1: system bus, DCode bus, ICode bus, CPU2: system bus, DMA1-bus, DMA2-bus and Radio system-bus) and ten slaves (3 x Flash memory, SRAM1, SRAM2a (backup), SRAM2b (non backup), AHB1 (including APB1 and APB2), AHB2, QUADSPI, AHB4, and AHB5).

### AHB/APB bridges

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

The bridges AHB to APB3 provide an a-synchronous connections between the AHB and the APB bus, allowing flexible selection of the frequency between the AHB and peripheral.

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

After each device reset, all peripheral clocks are disabled (except for the SRAM1/2 and Flash memory interface). Before using a peripheral you have to enable its clock in the RCC\_AHBxENR and the RCC\_APBxENR registers.

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

## 2.2 Memory organization

### 2.2.1 Introduction

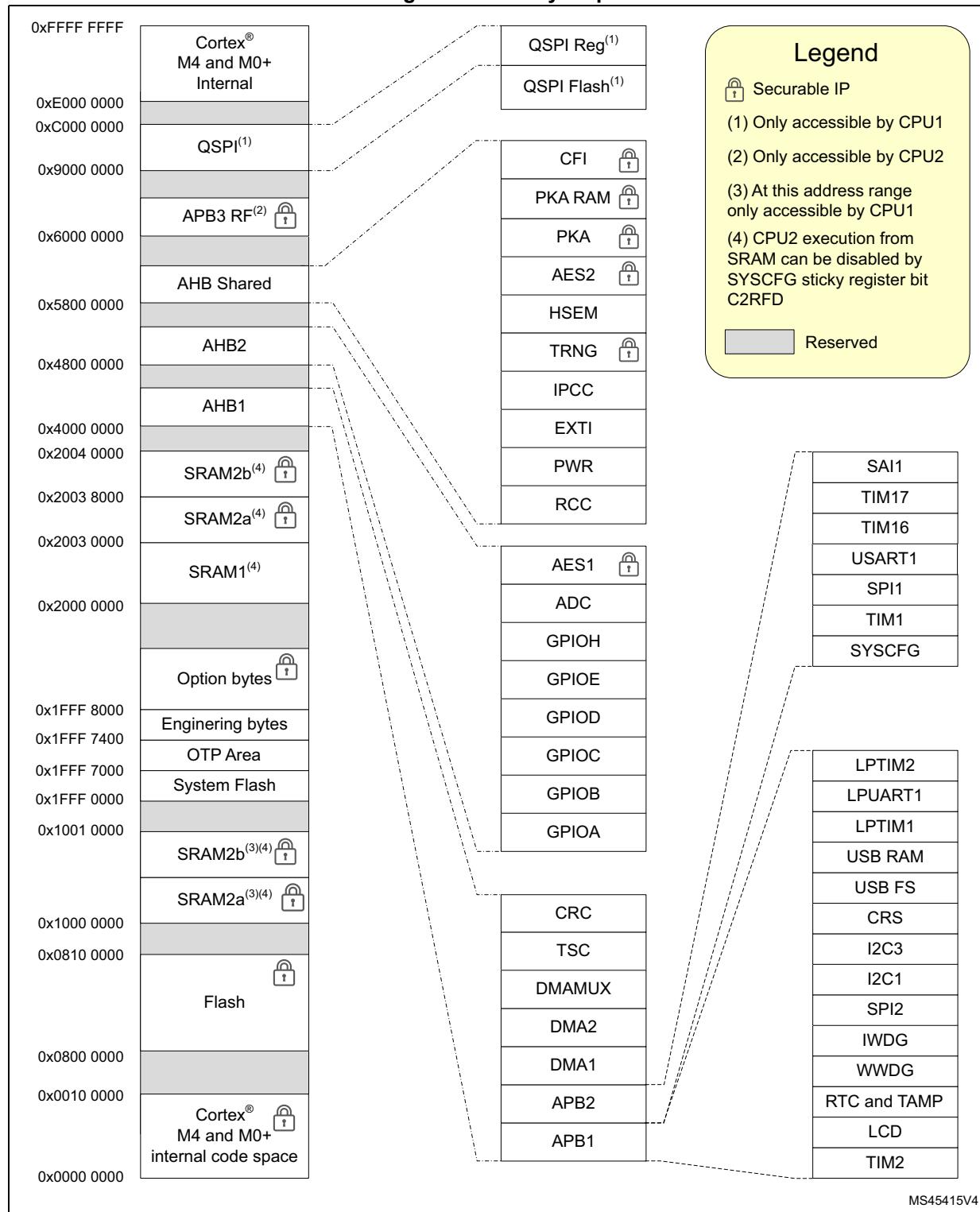
Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

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

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

## 2.2.2 Memory map and register boundary addresses

Figure 2. Memory map



1. It is forbidden to access QUADSPI Flash bank area before having properly configured and enabled the QUADSPI peripheral.

All the memory areas not allocated to on-chip memories and peripherals are considered “Reserved”. For the detailed mapping of available memory and register areas, refer to the following table, which gives the boundary addresses of the available peripherals.

**Table 1. Memory map and peripheral register boundary addresses**

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB3	0xA000 1000 - 0xA000 13FF	1 K	QUADSPI	<a href="#">Section 15.5.14: QUADSPI register map on page 414</a>
	0xA000 0000 - 0xA000 0FFF	4 K	Reserved	-
	0x9000 0000 - 0x9FFF FFFF	256 M	QUADSPI Flash	-
-	0x6000 2000 - 0x8FFF FFFF	-	Reserved	-
APB3	0x6000 1000 - 0x6000 1FFF	4 K	802.15.4 CTRL	-
	0x6000 0800 - 0x6000 0FFF	2 K	Reserved	-
	0x6000 0400 - 0x6000 07FF	1 K	Radio CTRL	-
	0x6000 0000 - 0x6000 03FF	1 K	BLE CTRL	-
	0x5800 4400 - 0x5FFF FFFF	128 K	Reserved	-
AHB4	0x5800 4000 - 0x5800 43FF	1 K	FLASH	<a href="#">Section 3.10.20: FLASH register map on page 128</a>
	0x5800 3400 - 0x5800 3FFF	3 K	Reserved	-
	0x5800 2400 - 0x5800 33FF	5 K	PKA RAM	<a href="#">Section 23.7.5: PKA register map on page 660</a>
	0x5800 2000 - 0x5800 23FF		PKA	
	0x5800 1C00 - 0x5800 1FFF	1 K	Reserved	-
	0x5800 1800 - 0x5800 1BFF	1 K	AES2	<a href="#">Section 22.7.18: AES register map on page 633</a>
	0x5800 1400 - 0x5800 17FF	1 K	HSEM	<a href="#">Section 38.4.9: HSEM register map on page 1303</a>
	0x5800 1000 - 0x5800 13FF	1 K	True RNG	<a href="#">Section 21.7.4: RNG register map on page 585</a>
	0x5800 0C00 - 0x5800 0FFF	1 K	IPCC	<a href="#">Section 37.4.9: IPCC register map on page 1290</a>
	0x5800 0800 - 0x5800 0BFF	1 K	EXTI	<a href="#">Section 14.6.17: EXTI register map on page 388</a>
	0x5800 0400 - 0x5800 07FF	1 K	PWR	<a href="#">Section 6.6.24: PWR register map and reset value table on page 192</a>
	0x5800 0000 - 0x5800 03FF	1 K	RCC	<a href="#">Section 8.4.49: RCC register map on page 283</a>

**Table 1. Memory map and peripheral register boundary addresses (continued)**

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB2	0x5006 0400 - 0x57FF FFFF	-	Reserved	-
	0x5006 0000 - 0x5006 03FF	1 K	AES1	<a href="#">Section 22.7.18: AES register map on page 633</a>
	0x5004 0000 - 0x5004 03FF	1 K	ADC	<a href="#">Section 16.9: ADC register map on page 504</a>
	0x4800 1C00 - 0x4800 1FFF	1 K	GPIOH	<a href="#">Section 9.5.12: GPIO register map on page 306</a>
	0x4800 1400 - 0x4800 1BFF	3 K	Reserved	-
	0x4800 1000 - 0x4800 13FF	1 K	GPIOE	<a href="#">Section 9.5.12: GPIO register map on page 306</a>
	0x4800 0C00 - 0x4800 0FFF	1 K	GPIOD	
	0x4800 0800 - 0x4800 0BFF	1 K	GPIOC	
	0x4800 0400 - 0x4800 07FF	1 K	GPIOB	
	0x4800 0000 - 0x4800 03FF	1 K	GPIOA	
AHB1	0x4002 4400 - 0x47FF FFFF	127 M	Reserved	-
	0x4002 4000 - 0x4002 43FF	1 K	TSC	<a href="#">Section 20.6.11: TSC register map on page 572</a>
	0x4002 3400 - 0x4002 3FFF	3 K	Reserved	-
	0x4002 3000 - 0x4002 33FF	1 K	CRC	<a href="#">Section 5.4.6: CRC register map on page 138</a>
	0x4002 0C00 - 0x4002 2FFF	9 K	Reserved	-
	0x4002 0800 - 0x4002 0BFF	1 K	DMAMUX	<a href="#">Section 12.6.7: DMAMUX register map on page 360</a>
	0x4002 0400 - 0x4002 07FF	1 K	DMA2	<a href="#">Section 11.6.7: DMA register map on page 343</a>
	0x4002 0000 - 0x4002 03FF	1 K	DMA1	
-	0x4001 5800 - 0x4001 FFFF	42 K	Reserved	-

**Table 1. Memory map and peripheral register boundary addresses (continued)**

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB2	0x4001 5400 - 0x4001 57FF	1 K	SAI1	<a href="#">Section 36.6.19: SAI register map on page 1275</a>
	0x4001 4C00 - 0x4001 53FF	2 K	Reserved	-
	0x4001 4800 - 0x4001 4BFF	1 K	TIM17	<a href="#">Section 26.4.23: TIM16/TIM17 register map on page 883</a>
	0x4001 4400 - 0x4001 47FF	1 K	TIM16	<a href="#">Section 26.4.23: TIM16/TIM17 register map on page 883</a>
	0x4001 3C00 - 0x4001 43FF	2 K	Reserved	-
	0x4001 3800 - 0x4001 3BFF	1 K	USART1	<a href="#">Section 33.8.15: USART register map on page 1121</a>
	0x4001 3400 - 0x4001 37FF	1 K	Reserved	-
	0x4001 3000 - 0x4001 33FF	1 K	SPI1	<a href="#">Section 35.6.8: SPI register map on page 1210</a>
	0x4001 2C00 - 0x4001 2FFF	1 K	TIM1	<a href="#">Section 24.4.30: TIM1 register map on page 759</a>
	0x4001 0400 - 0x4001 2BFF	10 K	Reserved	-
	0x4001 0200 - 0x4001 03FF	1 K	COMP	<a href="#">Section 18.6.3: COMP register map on page 523</a>
	0x4001 0100 - 0x4001 01FF		SYSCFG	<a href="#">Section 10.2.17: SYSCFG register map on page 323</a>
	0x4001 0030 - 0x4001 00FF		VREFBUF	<a href="#">Section 17.5.3: VREFBUF register map on page 509</a>
	0x4001 0000 - 0x4001 002F		SYSCFG	<a href="#">Section 10.2.17: SYSCFG register map on page 323</a>

**Table 1. Memory map and peripheral register boundary addresses (continued)**

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1	0x4000 9800 - 0x4000 FFFF	26 K	Reserved	-
	0x4000 9400 - 0x4000 97FF	1 K	LPTIM2	<a href="#">Section 27.7.11: LPTIM register map on page 909</a>
	0x4000 8400 - 0x4000 93FF	4 K	Reserved	-
	0x4000 8000 - 0x4000 83FF	1 K	LPUART1	<a href="#">Section 34.7.13: LPUART register map on page 1174</a>
	0x4000 7C00 - 0x4000 7FFF	1 K	LPTIM1	<a href="#">Section 27.7.11: LPTIM register map on page 909</a>
	0x4000 7000 - 0x4000 7BFF	3 K	Reserved	-
	0x4000 6C00 - 0x4000 6FFF	1K	USB SRAM	<a href="#">Section 39.6.3: USB register map on page 1336</a>
	0x4000 6800 - 0x4000 6BFF	1 K	USB FS	<a href="#">Section 39.6.3: USB register map on page 1336</a>
	0x4000 6400 - 0x4000 67FF	1 K	Reserved	-
	0x4000 6000 - 0x4000 63FF	1 K	CRS	<a href="#">Section 40.7.5: CRS register map on page 1347</a>
	0x4000 5C00 - 0x4000 5FFF	1 K	I2C3	<a href="#">Section 32.7.12: I2C register map on page 1034</a>
	0x4000 5800 - 0x4000 5BFF	1 K	Reserved	-
	0x4000 5400 - 0x4000 57FF	1 K	I2C1	<a href="#">Section 32.7.12: I2C register map on page 1034</a>
	0x4000 3C00 - 0x4000 53FF	6 K	Reserved	-
	0x4000 3800 - 0x4000 3BFF	1 K	SPI2	<a href="#">Section 35.6.8: SPI register map on page 1210</a>
	0x4000 3400 - 0x4000 37FF	1 K	Reserved	-
	0x4000 3000 - 0x4000 33FF	1 K	IWDG	<a href="#">Section 30.4.6: IWDG register map on page 961</a>
	0x4000 2C00 - 0x4000 2FFF	1 K	WWDG	<a href="#">Section 31.5.4: WWDG register map on page 967</a>
	0x4000 2800 - 0x4000 2BFF	1 K	RTC & TAMP	<a href="#">Section 29.7.21: RTC register map on page 951</a>
	0x4000 2400 - 0x4000 27FF	1 K	LCD	<a href="#">Section 19.6.6: LCD register map on page 553</a>
	0x4000 0400 - 0x4000 23FF	8 K	Reserved	-
	0x4000 0000 - 0x4000 03FF	1 K	TIM2	<a href="#">Section 25.4.25: TIMx register map on page 830</a>
AHB4	0x2003 8000 - 0x2003 FFFF	32 K	SRAM2b	-
	0x2003 0000 - 0x2003 7FFF	32 K	SRAM2a	-
	0x2000 3000 - 0x2002 FFFF	36 K	Reserved	-
AHB1	0x2000 0000 - 0x2002 FFFF	192 <sup>(1)</sup> K	SRAM1	-

**Table 1. Memory map and peripheral register boundary addresses (continued)**

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB4	0x1FFF 8000 - 0x1FFF 807F	128 B	Flash memory options	<a href="#">Section 3.10.20: FLASH register map on page 128</a>
	0x1FFF 7000 - 0x1FFF 73FF	1 K	Flash memory OTP	-
	0x1FFF 0000 - 0x1FFF 6FFF	28 K	Flash memory Boot loader	-
	0x1000 0000 - 0x1000 FFFF	64 K	SRAM2a/b CPU1 mirror	-
	0x0800 0000 - 0x080F FFFF	1 M <sup>(1)</sup>	User Flash memory	-
(2)	0x0000 0000 - 0x000F FFFF	1 M <sup>(1)</sup>	CPU1 Boot area	-

1. Depends upon device, see the product datasheet.

2. Bus depends upon selected CPU1 Boot area.

### 2.2.3 Bit banding

The CPU1 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.

The AHB1, APB1, APB2 peripheral registers and the SRAM1, SRAM2a and SRAM2b are mapped to a bit-band region, hence single bit-band write and read operations are allowed. The operations are only available for CPU1 accesses, and not from other bus masters (e.g. DMA)

The peripheral bit-band alias is located from address 0x4200 0000 to 0x42FF FFFF

The SRAM bit-band alias is located from address 0x2200 0000 to 0x227F FFFF

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

- bit\_word\_addr = bit\_band\_base + (byte\_offset \* 32) + (bit\_number \* 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 SRAM1 address 0x2000 0300 to the alias region.

$$0x2200\ 6008 = 0x2200\ 0000 + 0x0300 * 32 + 2 * 4$$

Writing to address 0x2200 6008 has the same effect as a read-modify-write operation on bit [2] of the byte at SRAM1 address 0x2000 0300.

Reading address 0x2200 6008 returns the value 0x01 or 0x00 of bit [2] of the byte at SRAM1 address 0x2000 0300.

For more information on bit-band, refer to the Cortex®-M4 programming manual.

## 2.3 Boot configuration

Three different CPU1 boot modes can be selected through the BOOT0 pin and nBOOT1 bit in the User options, as shown in [Table 2](#).

**Table 2. Boot modes**

nBOOT1 FLASH_OPTR[23]	nBOOT0 FLASH_OPTR[27]	BOOT0 pin PH3	nSWBOOT0 FLASH_OPTR[26]	Main Flash empty <sup>(1)</sup>	Boot memory space alias
x	x	0	1	0	Main Flash memory is selected as boot area
x	x	0	1	1	System memory is selected as boot area
x	1	x	0	x	Main Flash memory is selected as boot area
0	x	1	1	x	Embedded SRAM1 is selected as boot area
0	0	x	0	x	
1	x	1	1	x	System memory is selected as boot area
1	0	x	0	x	

1. A Flash empty check mechanism is implemented to force the boot from system Flash if the first Flash memory location is not programmed (0xFFFF FFFF) and if the boot selection was configured to boot from the main Flash.

The values on both BOOT0 and BOOT1 are latched after a reset. It is up to the user to provide the correct value for the required boot mode.

The BOOT0 and BOOT1 are also re-sampled when exiting Standby mode. Consequently they must be kept in the required boot mode. After the startup delay, the CPU1 fetches the top-of-stack from address 0x0000 0000, then starts code execution from the boot memory at 0x0000 0004.

Depending on the selected boot mode, main Flash, system Flash, or SRAM1 memories are accessible as follows:

- Boot from main Flash memory: the main Flash memory is aliased in the CPU1 boot memory space at address 0x0000 0000, and is also still accessible from its physical address 0x0800 0000. In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x0800 0000.
- Boot from System Flash memory: the system Flash memory is aliased in the CPU1 boot memory space at address 0x0000 0000, and is also still accessible from its physical address 0x1FFF 0000.
- Boot from SRAM memory: the SRAM is aliased in the CPU1 boot memory space at address 0x0000 0000, and is also still accessible from its physical address 0x2000 0000.

### CPU1 physical remap

Following CPU1 boot the application software can modify the memory map at address 0x0000 0000. This modification is performed by programming the [SYSCFG memory remap register \(SYSCFG\\_MEMRMP\)](#) in the SYSCFG controller.

The following memories can be remapped:

- Main Flash memory
- System Flash memory
- SRAM
- Quad SPI memory

### Embedded boot loader

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

- USART1 on pins PA9 and PA10
- I2C1 on pins PB6 and PB7.
- I2C3 on pins PC0 and PC1
- SPI1 on pins PA4, PA5, PA6 and PA7
- SPI2 on pins PB12, PB13, PB14 and PB15
- USB FS on pins PA11 and PA12.

## 2.4 CPU2 boot

Following a device reset the CPU2 will only boot after CPU1 has set the C2BOOT bit in the [PWR control register 4 \(PWR\\_CR4\)](#). The C2BOOT value is retained in Standby mode and the CPU2 will boot accordingly when exit from Standby.

The CPU2 will boot from its boot reset vector as defined by the Flash user Option C2OPT and SBRV.

The CPU2 may boot from anywhere in User Flash or SRAM1/SRAM2a/SRAM2b.

### CPU2 safe boot

When, after a reset, the User options are not valid and the BOOT0 and BOOT1 select CPU1 to boot from main Flash memory, the CPU2 will boot instead from a safe boot vector in main Flash memory at address 0x080F F000.

The safe boot can be used to restore the last known user options from a copied image.

## 2.5 CPU2 SRAM fetch disable

CPU2 execution from SRAM can be disabled by the C2RFD bit in SYSCFG register. Disabling CPU2 execution from SRAM improves robustness of the CPU2 software.

## 3 Embedded flash memory (FLASH)

### 3.1 Introduction

The flash memory interface manages CPU1 (CPU1 Cortex®-M4) AHB ICode and DCode accesses and the CPU2 (Cortex®-M0+) AHB to the flash memory. It implements the access arbitration between the two CPUs, the erase and program flash memory operations, the read and write protection, and the security mechanisms.

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

### 3.2 FLASH main features

- Up to 1 Mbyte of flash memory single bank architecture
- Memory organization: 1 bank
  - main memory: up to 1 Mbyte
  - page size: 4 Kbytes
- 72-bit wide data read (64 bits plus 8 ECC bits)
- 72-bit wide data write (64 bits plus 8 ECC bits)
- Page erase (4 Kbytes), and mass erase

Flash memory interface features:

- Flash memory read operations
- Flash memory program/erase operations
- Read protection activated by option (RDP)
- Two write protection areas selected by option (WRP)
- Two proprietary code read protection area selected by option (PCROP)
- CPU2 Security area
- Flash empty check
- Program and Erase suspension feature
- Prefetch on CPU1 ICODE and CPU2 S-bus
- CPU1 instruction cache: 32 cache lines of 4 x 64 bits on ICode (1 Kbyte RAM)
- CPU1 data cache: eight cache lines of 4 x 64 bits on DCode (256 bytes RAM)
- CPU2 instruction cache: four cache lines of 1 x 64 bits on S-bus (32 bytes RAM)
- Error code correction (ECC): eight bits for 64-bit
- Option byte loader

### 3.3 FLASH functional description

#### 3.3.1 Flash memory organization

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

The flash memory is organized as follows:

- A main memory block containing up to 256 pages for STM32WB55xx, up to 128 pages for STM32WB35xx of 4 Kbytes, each page with eight rows of 512 bytes.
- An information block containing:
  - System memory from which the CPU1 boots in System memory boot mode. The area is reserved and contains the boot loader used to reprogram the flash memory through one of the following interfaces: USART1, USB, I2C1, I2C3, SPI1, SPI2 (available only on STM32WB55xx). It is programmed by STMicroelectronics when the device is manufactured, and protected against spurious write/erase operations. For further details, refer to the AN2606 available from [www.st.com](http://www.st.com).
  - 1 Kbyte (128 double words) OTP (one-time programmable) for user data. The OTP data cannot be erased and can be written only once. If only one bit is at 0, the entire double word (64 bits) cannot be written anymore, even with the value 0x0000 0000 0000 0000.
  - Option bytes for user configuration.

The memory organization is based on a main area and an information block as shown in [Table 3](#).

**Table 3. Flash memory - Single bank organization**

Area	Addresses	Size (bytes)	Name
Main memory <sup>(1)</sup>	0x0800 0000 - 0x0800 0FFF	4 K	Page 0
	0x0800 1000 - 0x0800 1FFF	4 K	Page 1
	0x0800 2000 - 0x0800 2FFF	4 K	Page 2
	.	.	.
	.	.	.
	.	.	.
	0x0807 E000 - 0x0807 EFFF	4 K	Page 126
	0x0807 F000 - 0x0807 FFFF	4 K	Page 127
	0x0808 0000 - 0x0808 0FFF	4 K	Page 128
	.	.	.
	.	.	.
	.	.	.
Information block	0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
	0x1FFF 8000 - 0x1FFF 807F	128	Option bytes

1. Pages 128 to 255 available on STM32WB55xx only.

### 3.3.2 Empty check

During the OBL phase, after loading all options, the flash memory interface checks whether the first location of the main memory is programmed. The result of this check in conjunction with the boot0 and boot1 information is used to determine where the system has to boot

from. It prevents the system to boot from flash main memory area when, for instance, no user code has been programmed.

The flash main memory empty check status can be read from the EMPTY bit in *Flash memory access control register (FLASH\_ACR)*. Software can modify the flash main memory empty status by writing to the EMPTY bit.

### 3.3.3 Error code correction (ECC)

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

- One error detection and correction
- Two errors detection

When one error is detected and corrected, the flag ECCC (ECC correction) is set in *Flash memory ECC register (FLASH\_ECCR)*. If ECCCIE is set, an interrupt is generated.

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

When an ECC error is detected, the address of the failing double word is saved in ADDR\_ECC[16:0] in the FLASH\_ECCR register. ADDR\_ECC[2:0] are always cleared. The bus-ID of the CPU accessing the address is saved in CPUID[2:0].

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

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

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

### 3.3.4 Read access latency

To correctly read data from flash memory, the number of wait states (LATENCY) must be correctly programmed in the *Flash memory access control register (FLASH\_ACR)* according to the frequency of the flash memory (HCLK4) clock and the internal voltage range of the device  $V_{CORE}$ . Refer to [Section 6.1.6: Dynamic voltage scaling management](#).

*Table 4* shows the correspondence between wait states and flash clock frequency.

**Table 4. Number of wait states vs, Flash memory clock (HCLK4) frequency**

Wait states (WS) (LATENCY)	HCLK4 (MHz)	
	$V_{CORE}$ Range 1	$V_{CORE}$ Range 2
0 WS (1 HCLK cycles)	≤ 18	≤ 6
1 WS (2 HCLK cycles)	≤ 36	≤ 12
2 WS (3 HCLK cycles)	≤ 54	≤ 16
3 WS (4 HCLK cycles)	≤ 64	N/A

After POR on reset, the HCLK4 clock frequency is 4 MHz in Range 1 and 0 wait state (WS) is configured in the FLASH\_ACR register.

When woken-up from Standby, the HCLK4 clock frequency is 16 MHz in Range 1 and 0 wait state (WS) is configured in the FLASH\_ACR register.

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

### Increasing the CPU frequency

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

### Decreasing the CPU frequency

1. Modify the System clock source by writing the SW bits in the RCC\_CFGR register.
2. If needed, modify the flash memory clock prescaler by writing the SHDHPRE bits in RCC\_EXTCFGR.
3. Check that the new System clock source or/and the new flash memory clock prescaler value is/are taken into account by reading the clock source status (SWS bits) in the RCC\_CFGR register, or/and the AHB prescaler value (SHDHPREF bit), in the

- RCC\_EXTCFGR register, and wait until the programmed new system clock source or/and new flash memory clock prescaler value is/are read.
4. Program the new number of wait states to the LATENCY bits in *Flash memory access control register (FLASH\_ACR)*.
  5. Optionally, check that the new number of wait states is used to access the flash memory by reading back the LATENCY from the *Flash memory access control register (FLASH\_ACR)*.

### 3.3.5 Adaptive real-time memory accelerator (ART Accelerator)

The proprietary Adaptive real-time (ART) memory accelerator is optimized for STM32 industry-standard Arm® Cortex®-M4 with FPU processors. It balances the inherent performance advantage of the Arm® Cortex®-M4 with FPU over flash memory technologies, which normally require the processor to wait for the flash memory at higher operating frequencies.

To release the processor full performance, the accelerator implements an instruction prefetch queue and branch cache that increases program execution speed from the 64-bit flash memory. Based on CoreMark® benchmark, the performance achieved thanks to the ART Accelerator™ is equivalent to 0 wait state program execution from flash memory at a CPU frequency up to 64 MHz.

#### Instruction prefetch

The CPU1 fetches the instruction over the ICode bus and the literal pool (constant/data) over the DCode bus. The prefetch block aims at increasing the efficiency of ICode bus accesses.

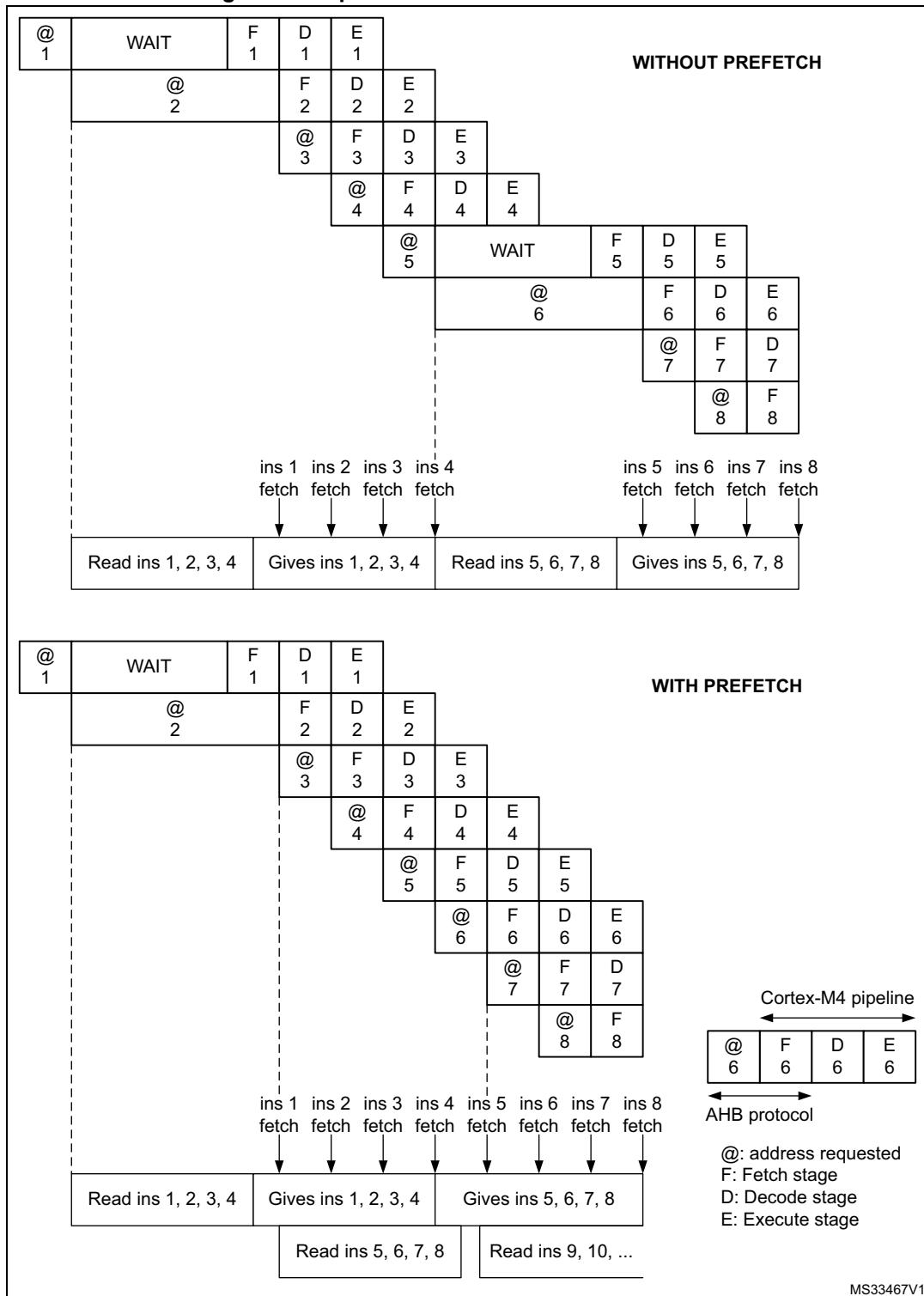
The CPU2 fetches the instruction and the literal pool (constant/data) over the S-bus. The prefetch block aims at increasing the efficiency of S-bus accesses.

Each flash memory read operation provides 64 bits from either two instructions of 32 bits or four instructions of 16 bits according to the program launched. This 64-bits current instruction line is saved in a current buffer. So, in case of sequential code, at least two CPU cycles are needed to execute the previous read instruction line. Prefetch on the CPU1 ICode bus or CPU2 S-bus can be used to read the next sequential instruction line from the flash memory while the current instruction line is being requested by the CPU.

Prefetch is enabled by setting the PRFTEN bit in the *Flash memory access control register (FLASH\_ACR)* for the CPU1 or *Flash memory CPU2 access control register (FLASH\_C2ACR)* for the CPU2. This feature is useful if at least one wait state is needed to access the flash memory.

*Figure 3* shows the execution of sequential 16-bit instructions with and without prefetch when three WS are needed to access the flash memory.

Figure 3. Sequential 16-bit instructions execution



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

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

### CPU1 Instruction cache memory (I-Cache)

To limit the CPU1 time lost due to jumps, it is possible to retain 32 lines of 4\*64 bits (1 Kbyte) in an instruction cache memory. This feature can be enabled for the CPU1 by setting the instruction cache enable (ICEN) bit in the *Flash memory access control register (FLASH\_ACR)*. Each time a miss occurs (requested data not present in the currently used instruction line, in the prefetched instruction line or in the instruction cache memory), the line read is copied into the instruction cache memory. If some data contained in the instruction cache memory are requested by the CPU, they are provided without inserting any delay. Once all the instruction cache memory lines have been filled, the LRU (least recently used) policy is used to determine the line to replace in the instruction memory cache. This feature is particularly useful in case of code containing loops.

The Instruction cache memory is enable after system reset.

### CPU1 Data cache memory (D-Cache)

CPU1 literal pools are fetched from flash memory through the DCode bus during the execution stage of the CPU pipeline. Each CPU1 DCode bus read access fetches 64 bits that are saved in a current buffer. The CPU pipeline is consequently stalled until the requested literal pool is provided. To limit the time lost due to literal pools, accesses through the AHB databus DCode have priority over accesses through the AHB instruction bus ICode.

If some literal pools are frequently used, the CPU1 data cache memory can be enabled by setting the data cache enable (DCEN) bit in the *Flash memory access control register (FLASH\_ACR)*. This feature works like the instruction cache memory, but the retained data size is limited to 8 lines of 4\*64 bits (256 bytes).

The Data cache memory is enabled after system reset.

*Note:*

*The D-Cache is active only when data is requested by the CPU (not by DMAs). Data in option bytes block are not cacheable.*

### CPU2 cache memory (S-bus)

To limit the CPU2 time lost due to jumps, it is possible to retain four lines of 1\*64 bits (32 bytes) in an instruction cache memory. This feature can be enabled for the CPU2 by setting the instruction cache enable (ICEN) bit in the *Flash memory CPU2 access control register (FLASH\_C2ACR)*. Each time a miss occurs (requested data not present in the currently used instruction line, in the prefetched instruction line or in the instruction cache memory), the line read is copied into the instruction cache memory. If some data contained in the instruction cache memory are requested by the CPU, they are provided without inserting any delay. Once all the instruction cache memory lines have been filled, the LRU (least recently used) policy is used to determine the line to replace in the instruction memory cache. This feature is particularly useful in case of code containing loops.

The Instruction cache memory is enable after system reset.

CPU2 literal pools are fetched from flash memory through the S-bus during the execution stage of the CPU pipeline. Each CPU2 S-bus read access fetches 64 bits that are saved in a current buffer. The CPU pipeline is consequently stalled until the requested literal pool is provided.

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

### 3.3.6 Flash memory program and erase operations

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

The **in-circuit programming (ICP)** method is used to update the entire contents of the flash memory, using the JTAG, SWD protocol or the supported interfaces by the System boot loader, to load the user application for both the CPU1 and CPU2, into the microcontroller. ICP offers quick and efficient design iterations and eliminates unnecessary package handling or socketing of devices.

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

The contents of the flash memory are not guaranteed if a device reset occurs during a flash memory operation.

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

*Note:* *In a multi-CPU system it is good practice to use semaphores to manage flash memory program and erase operations, and prevent simultaneous operations by the CPUs.*

#### Secure system flash memory programming

The secure CPU2 application can only be programmed by **in-application programming (IAP)** running on the secure CPU2. Only the secure CPU2 is able to download programming data into secure part of the memory. Secure IAP allows the user to re-program the flash memory while the application is running. Nevertheless, part of the application has to have been previously programmed in the flash memory using ICP.

The **in-circuit programming (ICP)** System boot loader is able to communicate with the secure CPU2 IAP, to download programming data into secure memory.

#### Unlocking the flash memory

After reset, write is not allowed in the [Flash memory control register \(FLASH\\_CR\)](#) or [Flash memory CPU2 control register \(FLASH\\_C2CR\)](#) to protect the flash memory against possible unwanted operations due, for example, to electric disturbances. The following sequence is used to unlock these registers:

1. Write KEY1 = 0x4567 0123 in the [Flash memory key register \(FLASH\\_KEYR\)](#)
2. Write KEY2 = 0xCDEF 89AB in the [Flash memory key register \(FLASH\\_KEYR\)](#).

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

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

*Note:* *The FLASH\_CR register cannot be written when the BSY bit in the [Flash memory status register \(FLASH\\_SR\)](#) or [Flash memory CPU2 status register \(FLASH\\_C2SR\)](#) is set. Any*

attempt to write to these registers with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

### 3.3.7 Flash main memory erase sequences

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

#### Flash memory page erase

The CPU1 is only able to page erase the non-secure part of the user flash.

The secure CPU2 is able to page erase both the secure and non-secure parts of the user flash memory.

A page erase starts only when allowed by the PESD bit in the *Flash memory status register (FLASH\_SR)* and in the *Flash memory CPU2 status register (FLASH\_C2SR)*.

When a page is protected by PCROP or WRP it is not erased.

Table 5. Page erase overview

Page	PCROP	WRP	PCROP_RDP	Comment	WRPERR	CPU1 bus error	CPU2 bus error
Non secure	No	No	x	Page is erased	No	No	No
		Yes		Page erase aborted (no page erase started)	Yes		
	Yes	No		Requested by CPU2, secure page is erased	No	N/A	No
		Yes		Requested by CPU1, secure page erase is aborted (no secure page erase started)		Yes	N/A
Secure	No	No		Page erase aborted (no page erase started)	Yes <sup>(1)</sup>		No
		Yes					
	Yes	No					
		Yes					

1. WRPERR is generated only when PER is requested by CPU2 (Cortex-M0+). There is no WRPERR when PER is requested by CPU1.

To erase a page (4 Kbytes), follow the procedure below:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the *Flash memory status register (FLASH\_SR)* or *Flash memory CPU2 status register (FLASH\_C2SR)*.

Check that flash memory program and erase operations are allowed by checking the PESD bit in the *Flash memory status register (FLASH\_SR)* or *Flash memory CPU2 status register (FLASH\_C2SR)* (these checks are recommended even if status may

- change due to flash memory operation requests by the other CPU, to limit the risk of receiving a bus error when starting page erase).
2. Check and clear all error programming flags due to a previous programming. If not, PGSER is set.
  3. Set the PER bit and select the page to erase (PNB) in the *Flash memory control register (FLASH\_CR)* or *Flash memory CPU2 control register (FLASH\_C2CR)*.
  4. Set the STRT bit in the FLASH\_xxCR register.
  5. Wait for the BSY bit to be cleared in the FLASH\_xxSR register.

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

### Flash memory mass erase

A flash memory mass erase by the CPU1 is ignored and a bus error is generated.

When PCROP or WRP is enabled any flash memory mass erase is aborted and no erase started.

**Table 6. Mass erase overview**

PCROP	WRP	PCROP_RDP	Comment	WRPERR	CPU1 bus error	CPU2 bus error		
No	No	X	Requested by secure CPU2, flash memory is mass erased	No	N/A	No		
No	Yes		Requested by secure CPU2, mass erase aborted (no erase started)	Yes				
Yes	No							
Yes	Yes		Requested by CPU1, mass erase aborted (no erase started)	No	Yes	N/A		
X	X							

To perform a mass erase, follow the procedure below:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the *Flash memory status register (FLASH\_SR)* or *Flash memory CPU2 status register (FLASH\_C2SR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSER is set.
3. Set the MER bit in the *Flash memory control register (FLASH\_CR)* or *Flash memory CPU2 control register (FLASH\_C2CR)*.
4. Set the STRT bit in the FLASH\_xxCR register.
5. Wait for the BSY bit to be cleared in the FLASH\_xxSR register.

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

### 3.3.8 Flash main memory programming sequences

The flash memory is programmed 72 bits (a double word, 64 bits plus ECC, 8 bits) at a time.

Programming in a previously programmed double word is only allowed when programming an all 0 value. It is not allowed to program any other value in a previously programmed double word, any attempt sets PROGERR flag in the *Flash memory status register (FLASH\_SR)* or *Flash memory CPU2 status register (FLASH\_C2SR)*, except when programming an already programmed double word with an all 0 value.

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

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

Only the secure CPU2 is able to download programming data into the secure part of the memory. A flash memory programming by the CPU1 in the secure flash memory area is ignored and a bus error is generated.

## Standard programming

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

1. Check that no flash main memory operation is ongoing by checking the BSY bit in the *Flash memory status register (FLASH\_SR)* or *Flash memory CPU2 status register (FLASH\_C2SR)*.  
Check that flash memory program and erase operations are allowed by checking the PESD bit in the *Flash memory status register (FLASH\_SR)* or *Flash memory CPU2 status register (FLASH\_C2SR)* (these checks are recommended even if status may change due to flash memory operation requests by the other CPU, to limit the risk of receiving a bus error when starting programming).
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the PG bit in the *Flash memory control register (FLASH\_CR)* or *Flash memory CPU2 control register (FLASH\_C2CR)*.
4. Perform the data write operation at the desired memory address, inside main memory block or OTP area. Only double word (64 bits) can be programmed.
  - a) Write a first word in an address aligned with double word
  - b) Write the second word.
5. Wait until the BSY bit is cleared in the FLASH\_xxSR register.
6. Check that EOP flag is set in the FLASH\_xxSR register (meaning that the programming operation has succeed), and clear it by software.
7. Clear the PG bit in the FLASH\_xxSR register if there no more programming request anymore.

Note:

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

*If the user needs to program only one word, double word must be completed with the erase value 0xFFFF FFFF to launch automatically the programming.*

*ECC is calculated from the double word to program.*

## Fast programming

This mode allows to program a row, 64 double words (512 bytes) and to reduce the page programming time by eliminating the need for verifying the flash memory locations before they are programmed and to avoid rising and falling time of high voltage for each double word. During fast programming, the flash memory clock frequency (HCLK4) must be at least 8 MHz.

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

Fast row programming must be performed by executing software from SRAM and disabling interrupts when not relocating the CPU interrupt vector table. A read access from the CPU requesting row programming causes a bus error. A read from any other source (the other CPU or DMA) is stalled until the row programming finishes (standard double word programming does not cause a bus error to the requesting CPU, but stalls any read until standard programming finishes).

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

1. Perform a mass erase. If not, PGSERR is set.
2. Check that no flash main memory operation is ongoing by checking the BSY bit in the *Flash memory status register (FLASH\_SR)* or *Flash memory CPU2 status register (FLASH\_C2SR)*.  
Check that flash memory program and erase operation is allowed by checking the PESD bit in the *Flash memory status register (FLASH\_SR)* or *Flash memory CPU2 status register (FLASH\_C2SR)* (these checks are recommended even if status may change due to flash memory operation requests by the other CPU, to limit the risk of receiving a bus error when starting programming).
3. Check and clear all error programming flag due to a previous programming.
4. Set the FSTPG bit in *Flash memory control register (FLASH\_CR)* or *Flash memory CPU2 control register (FLASH\_C2CR)*.
5. Write the 64 double words to program a row (512 bytes).
6. Wait until the BSY bit is cleared in the FLASH\_xxSR register.
7. Check that EOP flag is set in the FLASH\_xxSR register (meaning that the programming operation has succeed), and clear it by software.
8. Clear the FSTPG bit in the FLASH\_xxSR register if there are no more programming requests anymore.

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

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

*The 64 double word must be written successively. The high voltage is kept on the flash memory for all the programming. Maximum time between two double words write requests is the time programming (around 20  $\mu$ s). If a second double word arrives after this time programming, fast programming is interrupted and MISSER is set.*

*High voltage must not exceed 8 ms for a full row between two erases. This is guaranteed by the sequence of 64 double words successively written with a clock system greater or equal*

to 8 MHz. An internal time-out counter counts 7 ms when fast programming is set and stops the programming when time-out is over. In this case the FASTERR bit is set.

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

### Programming errors signaled by flags

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

- **PROGERR:** Programming error

In standard programming: PROGERR is set if the word to write with values different from all 0 has not been previously erased, except if the value to program is all 0. If any other error (SIZERR, PAGERR, PGSERR, WRPERR) occurs the PROGRERR may not be set even if there is a word re-programming without erase error.

- **SIZERR:** Size programming error

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

- **PGAERR:** Alignment programming error

PGAERR is set if one of the following conditions occurs:

- Alignment programming errors are individually checked per CPU. No checks are available for simultaneous multi-CPU programming. HSEM or other software mechanisms shall be used to prevent simultaneous multi CPU programming.
- In standard programming: the first word to be programmed is not aligned with a double word address, or the second word does not belong to the same double word address.
- In fast programming: the data to program does not belong to the same row than the previous programmed double words, or the address to program is not greater than the previous one.

- **PGSERR:** Programming sequence error

PGSERR is set if one of the following conditions occurs:

- In the standard programming sequence or the fast programming sequence: data is written when PG and FSTPG are cleared.
- In the standard programming sequence or the fast programming sequence: MER and PER are not cleared when PG or FSTPG is set.
- In the fast programming sequence: the Mass erase is not performed before setting the FSTPG bit.
- In the mass erase sequence: PG, FSTPG, and PER are not cleared when MER is set.
- In the page erase sequence: PG, FSTPG and MER are not cleared when PER is set.
- PGSERR is set also if PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, FASTERR or PGSERR is set due to a previous programming error.
- In the fast programming sequence: if the row to be programmed is on a page different from the one previously erased. If a mass erase has been done it is allowed to program rows on higher order pages, but not on lower order.

- **WRPERR:** Write protection error

WRPERR is set if one of the following conditions occurs:

- Attempt to program or erase in a write protected area (WRP) or in a PCROP area.
- Attempt to perform a mass erase when one page or more is protected by WRP or PCROP.
- The debug features are connected or the boot is executed from SRAM or from system flash when the read protection (RDP) is set to Level 1.
- Attempt to modify the option bytes when the read protection (RDP) is set to Level 2, except when requested by the secure CPU2.
- **MISSERR:** Fast programming data miss error  
In fast programming: all the data must be written successively. MISSERR is set if the previous data programmation is finished and the next data to program is not written yet.
- **FASTERR:** Fast programming error  
In fast programming: FASTERR is set if one of the following conditions occurs:
  - When FSTPG bit is set for more than 7  $\mu$ s, which generates a time-out detection.
  - When the row fast programming has been interrupted by a MISSERR, PGAERR, WRPERR or SIZERR.

If an error occurs during a program or erase operation, one of the following error flags is set in the *Flash memory status register (FLASH\_SR)* or *Flash memory CPU2 status register (FLASH\_C2SR)*:

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

In this case, if the error interrupt enable bit ERRIE is set in the *Flash memory control register (FLASH\_CR)* or in the *Flash memory CPU2 control register (FLASH\_C2CR)*, an interrupt is generated and the operation error flag OPERR is set in the FLASH\_xxSR register.

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

### Programming errors causing a bus error

Some error conditions, listed below, do not generate an error flag but a bus error instead.

- AHB write to any page when RDP level 1 and boot is performed from system flash memory or SRAM1
- AHB write when flash memory is powered down
- Read or write from flash memory through the debugger
- Reading from flash memory when fast row programming is ongoing, from the source which requested the fast row programming.
- Requesting a new programming request, when the previous one has not finished.
- FLASH\_CR register write between the two accesses of a double word programming.
- FLASH\_CR register write when PESDx is active (set).
- Writing a wrong key in FLASH\_KEYR or FLASH\_OPTKEYR register
- Any subsequent write to FLASH\_KEYR or FLASH\_OPTKEYR after unlocking the respective feature

### PGSERR and PGAERR in a page-based row programming

When performing a fast programming [Table 7](#) describes how PGSERR and PGAERR are handled.

**Table 7. Errors in page-based row programming**

Last page / row	Current page / row	MER active	PER active
page [x] / row [y]	page [x] / row [y-n]	PGAERR	PGAERR
page [x] / row [y]	page [x-n] / row [any]	PGAERR and PGSERR	PGAERR and PGSERR
page [x] / row [y]	page [x+n] / row [any]	No error	PGAERR

When after a system reset neither MER nor PER is performed, any programming attempt causes PGAERR and PGSERR errors.

### Programming and caches

If a flash memory write access impacts data in the data cache, the flash memory write access modifies the data in the memory and the data in the cache.

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

*Note:* *The I/D cache must be flushed only when it is disabled (I/DCEN = 0).*

## 3.4 FLASH option bytes

### 3.4.1 Option bytes description

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

A double word is split up in option bytes as indicated in [Table 8](#).

**Table 8. Option bytes format**

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

The organization of these bytes in the information block is shown in [Table 9](#). The option bytes can be read from the memory locations listed in [Table 9](#) or from the Option byte registers:

- *Flash memory option register (FLASH\_OPTR)*
  - *Flash memory PCROP zone A start address register (FLASH\_PCROP1ASR)*
  - *Flash memory PCROP zone A end address register (FLASH\_PCROP1AER)*
  - *Flash memory PCROP zone B start address register (FLASH\_PCROP1BSR)*
  - *Flash memory PCROP zone B end address register (FLASH\_PCROP1BER)*
  - *Flash memory WRP area A address register (FLASH\_WRP1AR)*
  - *Flash memory WRP area B address register (FLASH\_WRP1BR)*
  - *Flash memory IPCC mailbox data buffer address register (FLASH\_IPCCBR)*
  - *Secure flash memory start address register (FLASH\_SFR)*
  - *Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH\_SRRVR)*

**Table 9. Option bytes organization**

Table 9. Option bytes organization (continued)

Address <sup>(1)</sup>	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
1FFF8028	Unused																						PCROP1B_STRT									
1FFF8030	Unused																						PCROP1B_END									
1FFF8038 to 1FFF8060	Unused																															
1FFF8068	Unused																							IPCCDBA								
1FFF8070	Unused																							DDS	Unused	FSD	SFSA					
1FFF8078	C2OPT	NBRSD	SNBRSA			Unused	BRSRSD	SBRSA			SBRV																					
1FFF8FF8	OPVAL																															

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

### User and read protection option bytes

Flash memory address: 0x1FFF 8000

Reset value: 0b001x 1101 1xxx 1111 x111 0001 1010 1010 (ST production value)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
AGC_TRIM[2:0]			Res.	nBOOT0	nSW_BOOT0	SRAM2_RST	SRAM2_PE	nBOOT1	Res.	Res.	Res.	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW	Res.	nRST_SHDW	nRST_STDBY	nRST_STOP	BORLEV[2:0]																		
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								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Res.	nRST_SHDW	nRST_STDBY	nRST_STOP	ESE										RDP[7:0]								
	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 31:29 **AGC\_TRIM[2:0]**: Automatic gain control trimming.

Default value 0b001.

Bit 28 Reserved, must be kept at reset value.

Bit 27 **nBOOT0**: nBOOT0 option bit

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

Bit 26 **nSWBOOT0**: Software BOOT0

- 0: BOOT0 taken from the option bit nBOOT0
- 1: BOOT0 taken from PH3/BOOT0 pin

Bit 25 **SRAM2\_RST**: SRAM2 and PKA RAM erase when system reset

- 0: SRAM2 and PKA RAM erased when a system reset occurs
- 1: SRAM2 and non-secure PKA RAM not erased when a system reset occurs

Bit 24 **SRAM2\_PE**: SRAM2 parity check enable

- 0: SRAM2 parity check enable
- 1: SRAM2 parity check disable

Bit 23 **nBOOT1:** Boot configuration

Together with the BOOT0 pin or option bit nBOOT0 (depending on nSWBOOT0 option bit configuration), this bit selects boot mode from the flash main memory, SRAM1 or the System memory. Refer to [Section 2.3: Boot configuration](#).

Bits 22:20 Reserved, must be kept at reset value.

Bit 19 **WWDG\_SW:** Window watchdog selection

0: Hardware window watchdog  
1: Software window watchdog

Bit 18 **IWDG\_STDBY:** Independent watchdog counter freeze in Standby mode

0: Independent watchdog counter is frozen in Standby mode  
1: Independent watchdog counter is running in Standby mode

Bit 17 **IWDG\_STOP:** Independent watchdog counter freeze in Stop mode

0: Independent watchdog counter is frozen in Stop mode  
1: Independent watchdog counter is running in Stop mode

Bit 16 **IWDG\_SW:** Independent watchdog selection

0: Hardware independent watchdog  
1: Software independent watchdog

Bit 15 Reserved, must be kept at reset value.

Bit 14 **nRST\_SHDW:**

0: Reset generated when entering the Shutdown mode  
1: No reset generated when entering the Shutdown mode

Bit 13 **nRST\_STDBY:**

0: Reset generated when entering the Standby mode  
1: No reset generated when entering the Standby mode

Bit 12 **nRST\_STOP:**

0: Reset generated when entering the Stop mode  
1: No reset generated when entering the Stop mode

Bits 11:9 **BORLEV[2:0]:** BOR reset Level

These bits contain the VDD supply level threshold that activates/releases the reset.

000: BOR Level 0. Reset level threshold is ~ 1.7 V  
001: BOR Level 1. Reset level threshold is ~ 2.0 V  
010: BOR Level 2. Reset level threshold is ~ 2.2 V  
011: BOR Level 3. Reset level threshold is ~ 2.5 V  
100: BOR Level 4. Reset level threshold is ~ 2.8 V

Bit 8 **ESE:** System security enabled flag.

0: Security disabled  
1: Security enabled

Bits 7:0 **RDP[7:0]:** Read protection level

0xAA: Level 0, read protection not active  
0xCC: Level 2, chip read protection active  
Others: Level 1, memories read protection active

## PCROP1A start address option bytes

Flash memory address: 0x1FFF 8008

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxx1 1111 1111, (STM32WB55), 0xXXXX XXFF (STM32WB35), (ST production value)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
PCROP1A_STRT[8:0]															
								r	r	r	r	r	r	r	r

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

Bits 8:0 **PCROP1A\_STRT[8:0]**: PCROP1A area start offset

PCROP1A\_STRT contains the first 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB35xx devices.

### PCROP1A end address option bytes

Flash memory address: 0x1FFF 8010

Reset value: 0b1xxx xxxx xxxx xxxx xxxx xxx0 0000 0000 (STM32WB55), 0b1xxx xxxx xxxx xxxx xxxx xxxx 0000 0000 (STM32WB35) (ST production value)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCROP_RDP	Res.														
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
PCROP1A_END[8:0]															
							r	r	r	r	r	r	r	r	r

Bit 31 **PCROP\_RDP**: PCROP area preserved when RDP level is decreased

This bit is set only. It is reset after a full mass erase due to a change of RDP from Level 1 to Level 0.

- 0: PCROP area is not erased when the RDP level is decreased from Level 1 to Level 0.
- 1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:9 Reserved, must be kept at reset value.

Bits 8:0 **PCROP1A\_END**: PCROP1A area end offset

PCROP1A\_END contains the last 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB35xx devices.

### WRP Area A address option bytes

Flash memory address: 0x1FFF 8018

Reset value: 0xXX00 XXFF (STM32WB55) 0bxxxx xxxx x000 0000 xxxx xxxx x111 1111 (STM32WB35) (ST production value)

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **WRP1A\_END[7:0]**: WRP area “A” end offset

WRPA1\_END contains the last 4 Kbytes page of the WRP area “A”. Note that bit 23 is reserved on STM32WB35xx devices.

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

Bits 7:0 **WRP1A\_STRT[7:0]**: WRP area “A” start offset

WRPA1\_STRT contains the first 4 Kbytes page of the WRP area “A”. Note that bit 7 is reserved on STM32WB35xx devices.

### WRP Area B address option bytes

Flash memory address: 0x1FFF 8020

Reset value: 0xXX00 XXFF (STM32WB55), 0bxxxx xxxx x000 0000 xxxx xxxx x111 1111 (STM32WB35) (ST production value)

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **WRP1B\_END[7:0]**: WRP area “B” end offset

WRPB1\_END contains the last 4 Kbytes page of the WRP area “B”. Note that bit 23 is reserved on STM32WB35xx devices.

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

Bits 7:0 **WRP1B\_STRT[7:0]**: WRP area “B” start offset

WRPB1\_STRT contains the first 4 Kbytes page of the WRP area “B”. Note that bit 7 is reserved on STM32WB35xx devices.

### PCROP1B start address option bytes

Flash memory address: 0x1FFF 8028

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx1 1111 1111 (STM32WB55), 0xXXXX XXFF (STM32WB35) (ST production value)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PCROP1B_STRT[8:0]														
								r	r	r	r	r	r	r	r

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

Bits 8:0 **PCROP1B\_STRT[8:0]**: PCROP1B area start offset

PCROP1B\_STRT contains the first 2 Kbytes page of the PCROP1B area. Note that bit 8 is reserved on STM32WB35xx devices.

### PCROP1B end address option bytes

Flash memory address: 0x1FFF 8030

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxx0 0000 0000 (STM32WB55), 0xXXXX XX00 (STM32WB35) (ST production value)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PCROP1B_END[8:0]														
							r	r	r	r	r	r	r	r	r

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

Bits 8:0 **PCROP1B\_END[8:0]**: PCROP1B area end offset

PCROP1B\_END contains the last 2 Kbytes page of the PCROP1B area. Note that bit 8 is reserved on STM32WB35xx devices.

### IPCC mailbox data buffer address option bytes

Flash memory address: 0x1FFF 8068

Reset value: 0bXXXX XXXX XXXX XXXX XX00 0000 0000 0000 (ST production value)

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

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

Bits 13:0 **IPCCDBA[13:0]**: IPCC mailbox data buffer base address offset

IPCCDBA contains the first double-word offset of the IPCC mailbox data buffer area in SRAM2.

### Secure flash memory start address option bytes

Flash memory address: 0x1FFF 8070

Reset value: 0bxxxx xxxx xxxx xxxx xxx1 xxx0 xxxx xxxx (ST production value)

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

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **DDS**: Disable CPU2 debug access

- 0: CPU2 debug access enabled.
- 1: CPU2 debug access disabled.

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

Bit 8 **FSD**: Flash security disable

- FSD = 1: System and flash non-secure
- FSD = 0: System and flash secure. SFSA[7:0] contains the start address of the first 4 Kbytes page of the secure flash area.

Bits 7:0 **SFSA[7:0]**: Secure flash start address

System and flash secure. SFSA[7:0] contains the start address of the first 4 Kbytes page of the secure flash area.

### Secure SRAM2 start address and CPU2 reset vector option bytes

Flash memory address: 0x1FFF 8078

Reset value: 0xXXXX XXXX (ST production value)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C2OPT	NBRSD	SNBRSA[4:0]						Res.	BRSD	SBRSA[4:0]					
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
SBRV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **C2OPT**: CPU2 boot reset vector memory selection.

- 0: SBRV offset addresses SRAM1 or SRAM2, from start address 0x2000 0000 (SBRV value must be kept within the SRAM area).
- 1: SBRV offset addresses flash memory, from start address 0x0800 0000.

Bit 30 **NBRSD**: Non-backup SRAM2b security disable

- NBRSD = 1: SRAM2b is non-secure
- NBRSD = 0: SRAM2b is secure. SNBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area.

Bits 29:25 **SNBRSA**: Secure non-backup SRAM2b start address

NBRSD = 0: SRAM2b is secure. SNBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area.

Bit 24 Reserved, must be kept at reset value.

- Bit 23 **BRSD**: Backup SRAM2a security disable  
 BRSD = 1: SRAM2a is non-secure  
 BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.
- Bits 22:18 **SBRSA**: Secure backup SRAM2a start address  
 BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.
- Bits 17:0 **SBRV[17:0]**: CPU2 boot reset vector  
 Contains the word aligned CPU2 boot reset start address offset within the selected memory area by C2OPT.

### 3.4.2 Option bytes programming

After reset, the options related bits in the *Flash memory control register (FLASH\_CR)* and *Flash memory CPU2 control register (FLASH\_C2CR)* are write-protected. To run any operation on the option bytes page, the option lock bit OPTLOCK in the *Flash memory control register (FLASH\_CR)* must be cleared. The following sequence is used to unlock this register:

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

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

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

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

*Note:* In a multi-CPU system it is good practice to use semaphores to manage option programming, and prevent simultaneous option programming by the CPUs.

#### Modifying user options

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

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

1. Clear OPTLOCK option lock bit with the clearing sequence described above
2. Write the desired options value in the options registers.
3. Check that no flash memory operation is on going by checking the BSY bit in the *Flash memory status register (FLASH\_SR)* or *Flash memory CPU2 status register (FLASH\_C2SR)*.

Check that flash program and erase operation is allowed by checking the PESD bit in the *Flash memory status register (FLASH\_SR)* or *Flash memory CPU2 status register (FLASH\_C2SR)* (these checks are recommended even if status may change due to

flash operation requests by the other CPU, to limit the risk of receiving a bus error when modifying user options).

4. Set the Options start bit OPTSTRT in the [\*Flash memory control register \(FLASH\\_CR\)\*](#).
5. Wait for the BSY bit to be cleared.

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

### Secure user options

The secure option bytes [\*Secure flash memory start address register \(FLASH\\_SFR\)\*](#) and [\*Flash memory secure SRAM2 start address and CPU2 reset vector register \(FLASH\\_SRRVR\)\*](#) can only be written by the secure CPU2.

### Option byte loading

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

Option bytes loading is performed in two cases:

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

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

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

During option byte loading, the options are read by double word. ECC on option words is not taken into account during OBL, but only during direct SW read of option area.

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

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

- for USR OPT option, the value of mismatch is all options at ‘1’, except for BOR\_lev that is “000” (lowest threshold)
- for WRP option, the value of mismatch is the default value “No protection”
- for RDP option, the value of mismatch is the default value “Level 1”
- for PCROP, the value of mismatch is “all memory protected”
- for FSD and SFSA option, the value of mismatch is “all memories (Flash, SRAM2a and SRAM2b) secured”
- for BRSD, SBRSA, NBRSD and SNBRASA options, the value of mismatch is “SRAM2 memory part secured”
- for DDS option, the value of mismatch is “CPU2 debug disabled”
- for C2OPT and SBRV options, the value of mismatch is “CPU2 boot from start address of last Flash page (safe boot)”
- for OPTVAL option, the value of mismatch is “not valid”.

If the OPTVAL option indicates “not valid” all memories (Flash, SRAM2a and SRAM2b) are secure.

**Table 10. Option loading control**

OPTVERR	OPTNV	Description
0	0	Options correctly loaded and OPTVAL is “valid”. – Security applied according to options.
0	1	Does not occur
1	0	OPTVAL option correctly loaded as “valid”, but some or all other option and engineering bits corrupted, mismatch values loaded. – When secure option is loaded correctly, security is applied according to the loaded secure option values. – When secure option is corrupted, security is applied on the full memory as indicated by the loaded mismatch value.
1	1	Some or all option and engineering bits corrupted, mismatch values loaded. OPTVAL correctly loaded as “not valid”. Security applied on full memories irrespective of the loaded secure option values.

On system reset rising, internal option registers are copied into option registers that can be read and written by software:

- FLASH\_OPTR
- FLASH\_PCROPxySR (x = 1 and y = A or B)
- FLASH\_PCROPxyER (x = 1 and y = A or B)
- FLASH\_WRPxyR (x = 1 and y = A or B)
- FLASH\_IPCCDBA
- FLASH\_SFR
- FLASH\_SRRVA

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

### 3.5 FLASH UID64

A 64-bit unique device identification (UID64) is stored in the flash memory, and can be accessed by the CPUs.

**Table 11. UID64 organization**

Address	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
0x1FFF 7580																																
0x1FFF 7584																																

The UID64 is programmed at device production, and provides a unique code for each device

- the 24-bit ST company ID returns 0x00 80 E1
- the 8-bit device ID returns 0x26 for STM32WB55xx revision Y and STM32WB35xx revision A, 0x27 for STM32WB55xx/35xx revision X, as indicated in `DBGMCU_IDCODE`
- the 32-bit unique device number is a sequential number, different for each individual device.

The 64-bit UID64 may be used by firmware to derive BLE 48-bit device address EUI-48 or 802.15.4 64-bit device address EUI-64.

## 3.6 Flash memory protection

The flash main memory can be protected against external accesses with the Read protection (RDP). The pages can also be protected against unwanted write (WRP) due to loss of program counter context. The write-protection WRP granularity is 4 Kbytes. Apart from the RDP and WRP, flash memory can also be protected against read and write from third parties (PCROP). The PCROP granularity is 2 Kbytes.

Part of the flash main memory can be secured. It grants exclusive access to this part of the memory to the CPU2.

### 3.6.1 Read protection (RDP)

The read protection is activated by setting the RDP option byte and then, by applying a system reset to reload the new RDP option byte. The read protection protects the flash main memory, the option bytes, the backup registers (RTC\_BKPxR in the RTC) and the SRAM2.

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

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

The flash memory is protected when the RDP option byte and its complement contain the pair of values shown in [Table 12](#).

**Table 12. Flash memory read protection status**

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

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

#### Level 0: No protection

Read, program and erase operations into the flash main memory area are possible. The option bytes, the SRAM2 and the backup registers are also accessible by all operations.

## Level 1: Read protection

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

- **User mode:** Code executing in user mode (**Boot flash**) can access flash main memory, option bytes, SRAM2 and backup registers with all operations.
- **Debug, boot RAM and boot loader modes:** In debug mode or when code is running from boot RAM or boot loader, the flash main memory, the backup registers (RTC\_BKPxR in the RTC) and the SRAM2 are totally inaccessible. In these modes, a read or write access to the flash memory generates a bus error and a hard fault interrupt.

**Caution:** In case the Level 1 is configured and no PCROP areas are defined, it is mandatory to set PCROP\_RDP bit to 1 (full mass erase when the RDP level is decreased from Level 1 to Level 0). In case the Level 1 is configured and a PCROP area is defined, if user code needs to be protected by RDP but not by PCROP, it must not be placed in a page containing a PCROP area.

## Level 2: No debug

In this level, the protection Level 1 is guaranteed. In addition, the CPU1 and CPU2 debug port, the boot from RAM (boot RAM mode) and the boot from System memory (boot loader mode) are no more available. In user execution mode (boot FLASH mode), all operations are allowed on the flash main memory. On the contrary, only read and secure write operations can be performed on the option bytes. Option bytes, can only be programmed and erased by a secure CPU2.

The Level 2 cannot be removed from the non-secure application side: it is an irreversible operation. When attempting to modify the options bytes, the protection error flag WRPERR is set in the FLASH\_xxxSR register and an interrupt can be generated.

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

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

## Changing the Read protection level

It is easy to move from Level 0 to Level 1 by changing the value of the RDP byte to any value (except 0xCC). By programming the 0xCC value in the RDP byte, it is possible to go to level 2 either directly from Level 0 or from Level 1. Once in Level 2 it is no more possible to modify the Read protection level.

When the RDP is reprogrammed to the value 0xAA to move from Level 1 to Level 0, a mass erase of the flash main memory is performed if PCROP\_RDP is set in the [Flash memory PCROP zone A end address register \(FLASH\\_PCROP1AER\)](#). The backup registers (RTC\_BKPxR in the RTC), the SRAM2 and the PKA SRAM are also erased. The user options (except PCROP protection) are set to their previous values copied from FLASH\_OPTR, FLASH\_WRPxyR (x = 1 and y = A or B). PCROP is disabled. The OTP area is not affected by mass erase and remains unchanged.

If the bit PCROP\_RDP is cleared in the FLASH\_PCROP1AER, the full mass erase is replaced by a partial mass erase that is successive page erases, except for the pages protected by PCROP. This is done in order to keep the PCROP code. Only when the flash

memory is erased, options are re-programmed with their previous values. This is also true for FLASH\_PCROPxySR and FLASH\_PCROPxyER registers ( $x = 1$  and  $y = A$  or  $B$ ).

A requested mass erase performs a partial mass erase (a sequence of page erases), except for the pages protected by CPU2 security (SFSA). This is done to keep the CPU2 secure code.

**Table 13. RDP regression from Level 1 to Level 0 and memory erase**

SFSA	PCROP	PCROP_RDP	Comment
Partial	None	x	Flash memory multiple page erase of all non-secure pages, SRAM2 and PKA RAM and backup registers erase. (secure flash pages conserved).
	Partial	1	
		0	Flash memory multiple page erase of all non-PCROP pages and non-secure pages, SRAM2 and PKA RAM and backup registers erase (PCROP flash memory pages and secure flash memory pages conserved).
	Complete non-secure		Flash memory, SRAM2 and PKA RAM and backup registers are conserved.
Complete flash memory	x	x	Flash memory, SRAM2 and PKA RAM and backup registers are conserved.

**Note:** *Partial mass erase is performed only when Level 1 is active and Level 0 requested. When the protection level is increased (0→1, 1→2, 0→2) there is no mass erase.*

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

Figure 4. Changing the Read protection (RDP) level

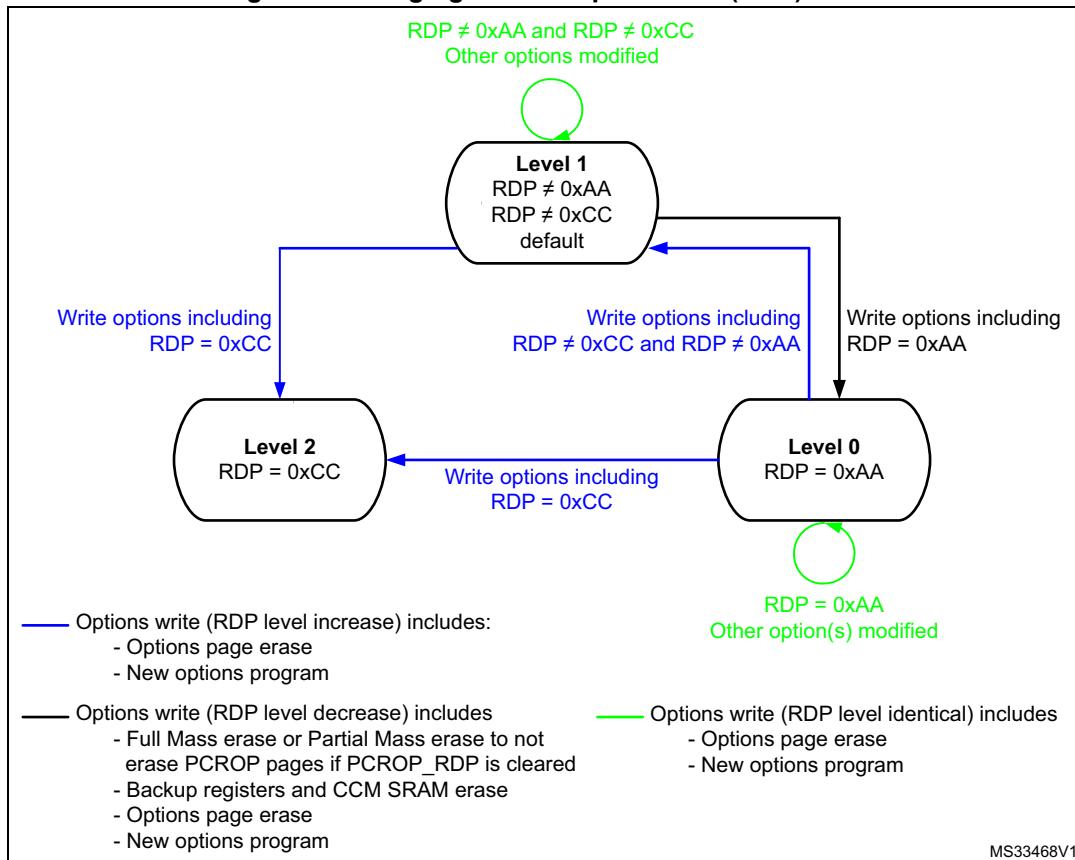


Table 14. Access status vs. protection level and execution modes

Area	Protection level	User execution (BootFromFlash memory)			Debug / BootFromRam / BootFromLoader		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	1	Yes	Yes	Yes	No	No	No <sup>(4)</sup>
	2	Yes	Yes	Yes	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>
System memory <sup>(2)</sup>	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>
Option bytes	1	Yes	Yes <sup>(3)</sup>	Yes	Yes	Yes <sup>(3)</sup>	Yes
	2	Yes	CPU1 and CPU2 none secure - No	CPU1 and CPU2 none secure - No	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>
			CPU2 secure - Yes	CPU2 secure - Yes			
Backup registers	1	Yes	Yes	N/A	No	No	No <sup>(4)</sup>
	2	Yes	Yes	N/A	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>
SRAM2	1	Yes	Yes	N/A	No	No	No <sup>(5)</sup>
	2	Yes	Yes	N/A	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>

- When the protection Level 2 is active, the Debug port, the boot from RAM and the boot from system memory are disabled.
- The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.

3. The flash non secure main memory is erased when the RDP option byte is programmed with all level of protections disabled (0xAA). The flash secure main memory is also erased when the SFSA option byte is programmed to be non-secure and at the same time the RDP option byte is programmed with all level protections disabled (0xAA).
4. The backup registers are erased when RDP changes from Level 1 to Level 0.
5. The SRAM2 is erased when RDP changes from Level 1 to Level 0.

### 3.6.2 Proprietary code readout protection (PCROP)

Two parts of the flash memory can be protected against read and write from third parties.

The protected area is execute-only: it can only be reached by the STM32 CPUs, with an instruction code, while all other accesses (DMA, debug and CPU data read, write and erase) are strictly prohibited. The PCROP areas have a 2 Kbytes granularity. An additional option bit (PCROP\_RDP) makes it possible to select if the PCROP area is erased or not when the RDP protection is changed from Level 1 to Level 0 (refer to [Changing the Read protection level](#)).

Each PCROP area is defined by a start page offset and an end page offset into the flash memory. These offsets are defined in the PCROP address registers *Flash memory PCROP zone A start address register (FLASH\_PCROP1ASR)*, *Flash memory PCROP zone A end address register (FLASH\_PCROP1AER)*, *Flash memory PCROP zone B start address register (FLASH\_PCROP1BSR)* and *Flash memory PCROP zone B end address register (FLASH\_PCROP1BER)*.

A PCROP area is defined from the address *Flash memory Base address + [PCROPxy\_STRT x 0x800]* (included) to the address: *Flash memory Base address + [(PCROPxy\_END+1) x 0x800]* (excluded). The minimum PCROP area size is two PCROP pages (4 Kbytes)  $PCROPxy\_END = PCROPxy\_STRT + 1$ .

When  $PCROPxy\_END = PCROPxy\_STRT$  the full flash memory is PCROP protected.

For example, to protect by PCROP from the address 0x0806 2F80 (included) to the address 0x0807 0004 (included):

- if boot in flash memory is selected, one of the FLASH\_PCROPxySR and FLASH\_PCROPxyER registers (x = 1 and y = A or B) must be programmed with:
  - $PCROPxy\_STRT = 0xC5$  (PCROP area first address 0x0806 2800)
  - $PCROPxy\_END = 0xE0$  (PCROP area last address 0x0807 07FF)

Any data read access performed through a PCROP protected area triggers the RDERR flag error.

Any PCROP protected address is also write protected and any write access to one of these addresses triggers WRPERR.

Any PCROP area is also erase protected. Consequently, any erase to a page in this zone is impossible (including the page containing the start address and the end address of this zone). Moreover, a software mass erase cannot be performed if one zone is PCROP protected.

In the previous example, due to erase by page, all pages from page 0xC5 to 0xE0 are protected in case of page erase, all addresses from 0x0806 2800 to 0x0807 07FF cannot be erased.

Deactivation of PCROP can only occur when the RDP is changing from Level 1 to Level 0. If the user options modification tries to clear PCROP or to decrease the PCROP areas, the

options programming is launched but PCROP areas stays unchanged. On the contrary, it is possible to increase the PCROP areas.

When option bit PCROP\_RDP is cleared, and when the RDP is changing from Level 1 to Level 0, Full mass erase is replaced by Partial mass erase to preserve the PCROP area (refer to [Changing the Read protection level](#)). In this case, PCROPxy\_STRT and PCROPxy\_END (x = 1 and y = A or B) are not erased.

**Table 15: PCROP protection**

PCROP registers values (x = 1 and y = A or B)	PCROP protection area
PCROPxy_STRT = PCROPxy_END	The full flash memory is PCROP protected
PCROPxy_STRT > PCROPxy_END	No PCROPxy, unprotected
PCROPxy_STRT < PCROPxy_END	Pages from PCROPxy_STRT to PCROPxy_END are protected

**Note:** *It is recommended to align PCROP areas with page granularity when using PCROP\_RDP, or to leave free the rest of the page where PCROP zones start or end.*

### 3.6.3 Write protection (WRP)

The user area in flash memory can be protected against unwanted write operations. Two write-protected (WRP) areas can be defined, with page (4 Kbytes) granularity. Each area is defined by a start page offset and an end page offset related to the physical flash memory base address. These offsets are defined in the WRP address registers [Flash memory WRP area A address register \(FLASH\\_WRP1AR\)](#) and [Flash memory WRP area B address register \(FLASH\\_WRP1BR\)](#).

The WRP “y” area (y = A, B) is defined from the address *Flash memory Base address + [WRP1y\_STRT x 0x1000]* (included) to the address *Flash memory Base address + [(WRP1y\_END+1) x 0x1000]* (excluded). The minimum WRP area size is one WRP page (4 Kbytes), *WRP1y\_END = WRP1y\_STRT*.

For example, to protect by WRP from the address 0x0806 2000 (included) to the address 0x0807 1FFF (included):

- If boot in flash memory is selected, FLASH\_WRP1AR register must be programmed with:
  - WRP1A\_STRT = 0x62.
  - WRP1A\_END = 0x71.

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

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

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

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

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

**Table 16: WRP protection**

WRP registers values (x = 1 and y = A or B)	WRP protection area
WRPx <sub>y</sub> _STRT = WRPx <sub>y</sub> _END	Page WRPx <sub>y</sub> is protected
WRPx <sub>y</sub> _STRT > WRPx <sub>y</sub> _END	No WRP, unprotected
WRPx <sub>y</sub> _STRT < WRPx <sub>y</sub> _END	Pages from WRPx <sub>y</sub> _STRT to WRPx <sub>y</sub> _END are protected

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

### 3.6.4 CPU2 security (ESE)

All or a part of the flash memory and the SRAM2a and SRAM2b can be made secure, exclusively accessible by the CPU2, protected against execution, read and write from third parties. Only the CPU2 can execute, read and write in these areas. It can only be reached by the CPU2, while all other accesses (CPU1 and DMA) are strictly prohibited.

#### Changing the CPU2 security mode

CPU2 security start address can be modified by the secure CPU2 by loading a new user option SFSA.

#### CPU2 secure flash memory area

The CPU2 secure flash memory area has sector (4 Kbytes) granularity and is defined by the secure flash memory start page offset user option (SFSA) into the flash memory. This offset is controlled from the SFSA field in the [Secure flash memory start address register \(FLASH\\_SFR\)](#).

The CPU2 secure flash memory area is defined as follows: *Flash memory Base address + [SFSA x 0x1000] (included)* to the last flash memory address: When CPU2 security is enabled, the minimum CPU2 secure area size is one sector (4 Kbytes).

For example, a CPU2 secure area from the address 0x080E 7000 (included) to the address 0x080F FFFF (included):

- FLASH\_SFR registers must be programmed with:
  - SFSA= 0x0E7.

A flag (ESE) is available from the [Flash memory option register \(FLASH\\_OPTR\)](#) informing that CPU2 security is enabled.

Any CPU1 access to a CPU2 security area triggers RDERR or WRPERR flag error.

#### CPU2 secure SRAM2 areas

The CPU2 secure SRAM2a and SRAM2b areas have a 1 Kbyte granularity and are defined by the secure backup RAM (SRAM2a) start address user options (BRSD and SBRSA) and the secure non-backup RAM (SRAM2b) start address user options (NBRSD and SNBRSA) into the flash memory. These offset are controlled by the SBRSA and SNBRSA fields in the

*Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH\_SRRVR).*

The CPU2 secure SRAM2a area is defined as *Backup SRAM2a Base address + [SBRSA x 0x0400]* (included) to the last SRAM2a address.

As an example, for a CPU2 secure SRAM2a area from the address 0x2003 5000 (included) to the address 0x2003 7FFF (included) FLASH\_SRRVR registers must be programmed with SBRSA = 0x14.

Any CPU1 read access returns no data, and a write access to a CPU2 security SRAM2a area is discarded and triggers a bus error.

When BRSD is set to 1 the SRAM2a is non-secure.

The CPU2 secure non-backup SRAM2b area is defined as *Non-backup SRAM2b base address + [SNBRSA x 0x0400]* (included) to the last SRAM2b address.

As an example, for a CPU2 secure SRAM2b area from the address 0x2003 EC00 (included) to the address 0x2003 FFFF (included) FLASH\_SRRVR registers must be programmed with SNBRSA = 0x1B.

Any CPU1 read access returns no data, and a write access to a CPU2 security SRAM2b area is discarded and triggers a bus error.

When NBRSD is set to 1 the SRAM2b is non-secure.

### CPU2 debug access

Debug access to the CPU2 is disabled, as indicated by the DDS filed in the *Secure flash memory start address register (FLASH\_SFR)*. The debugger has no access to the CPU2 and the secure peripherals and memory areas.

## 3.7 FLASH program/erase suspension

Flash memory program and erase operations can be suspended by setting the PES bit in the *Flash memory access control register (FLASH\_ACR)* or *Flash memory CPU2 access control register (FLASH\_C2ACR)*. This feature is useful when executing time critical sections by a CPU. It makes it possible to suspend any new program or erase operation from being started, preventing CPU instruction and data fetches from being blocked.

When at least one PES bit is set:

- Any ongoing program or erase operation is completed.
  - The maximum latency for a flash memory program/erase suspension is the maximum time for one program or erase operation to complete (see the device datasheet for more information on the program and erase timing).
- All new requested program and erase operations are not started, but suspended.
  - PESD bits in the *Flash memory status register (FLASH\_SR)* and *Flash memory CPU2 status register (FLASH\_C2SR)* are set as soon as any PES is set, no matter if a program/erase is currently suspended. This allows a CPU to test PESD prior to requesting a program or an erase operation.

When all PES bits are reset to 0:

- A suspended program or erase operation is started.
  - The PESD bit in both the *Flash memory status register (FLASH\_SR)* and in the *Flash memory CPU2 status register (FLASH\_C2SR)* are cleared.

## 3.8 FLASH interrupts

Table 17. Flash memory interrupt requests

Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit
End of operation	EOP <sup>(1)</sup>	Write EOP = 1	EOPIE
Operation error	OPERR <sup>(2)</sup>	Write OPERR = 1	ERRIE
Read protection error	RDERR	Write RDERR = 1	RDERRIE
Write protection error	WRPERR	Write WRPERR = 1	N/A
Size error	SIZERR	Write SIZERR = 1	N/A
Programming sequential error	PROGERR	Write PROGERR = 1	N/A
Programming alignment error	PGAERR	Write PGAERR = 1	N/A
Programming sequence error	PGSERR	Write PGSERR = 1	N/A
Data miss during fast programming error	MISSERR	Write MISSERR = 1	N/A
Fast programming error	FASTERR	Write FASTERR = 1	N/A
ECC error correction	ECCC	Write ECCC = 1	ECCCIE
ECC double error (NMI)	ECCD	Write ECCD = 1	N/A

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

**Note:** *The flash interface provides only a single interrupt line to both CPUs. It is good practice to manage flash operations between the CPUs using a semaphore. To avoid receiving unwanted flash operation interrupts, the CPU must mask the flash interrupt in the NVIC or SYSCFG pre-mask.*

## 3.9 Register access protection

The user options registers can be protected by security.

The FLASH secure registers (FSD, SFSA, BRSD, SBRSA, NBRSD, SNBRSA, SBRV, C2OPT, and DDS) are secure and can be written only by the secure CPU2 and read by any CPU, secure and non-secure. When the CPU1 tries to write, the write is discarded.

## 3.10 FLASH registers

### 3.10.1 Flash memory access control register (FLASH\_ACR)

Address offset: 0x000

Reset value: 0x0000 0600

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

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

Bit 16 **EMPTY**: Flash memory user area empty.

When read indicates whether the first location of the user flash memory is erased or has a programmed value.

0: Read: User flash memory programmed

1: Read: User flash memory empty

When written this bit is overwritten with the written value.

Bit 15 **PES**: CPU1 program / erase suspend request

0: Flash memory program and erase operations granted.

1: Any new flash memory program and erase operation is suspended until this bit and the same bit in *Flash memory CPU2 access control register (FLASH\_C2ACR)* are cleared. The PESD bit in both the *Flash memory status register (FLASH\_SR)* and *Flash memory CPU2 status register (FLASH\_C2SR)* is set when at least one PES bit in FLASH\_ACR or FLASH\_C2ACR is set.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **DCRST**: CPU1 Data cache reset

0: CPU1 Data cache is not reset

1: CPU1 Data cache is reset

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

Bit 11 **ICRST**: CPU1 Instruction cache reset

0: CPU1 Instruction cache is not reset

1: CPU1 Instruction cache is reset

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

Bit 10 **DCEN**: CPU1 Data cache enable

0: CPU1 Data cache is disabled

1: CPU1 Data cache is enabled

Bit 9 **ICEN**: CPU1 Instruction cache enable

0: CPU1 Instruction cache is disabled

1: CPU1 Instruction cache is enabled

Bit 8 **PRFTEN**: CPU1 Prefetch enable

0: CPU1 Prefetch disabled

1: CPU1 Prefetch enabled

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

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

These bits represent the ratio of the flash memory HCLK clock period to the flash memory access time.

- 000: Zero wait states
- 001: One wait state
- 010: Two wait states
- 011: Three wait states
- Others: Reserved

### 3.10.2 Flash memory key register (FLASH\_KEYR)

Address offset: 0x008

Reset value: 0x0000 0000

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

Bits 31:0 **KEY[31:0]**: Flash memory key

The following values must be written consecutively to unlock the *Flash memory control register (FLASH\_CR)* and *Flash memory CPU2 control register (FLASH\_C2CR)*, thus enabling programming/erasing operations:

KEY1: 0x4567 0123

KEY2: 0xCDEF 89AB

### 3.10.3 Flash memory option key register (FLASH\_OPTKEYR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

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

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

KEY1: 0x0819 2A3B

KEY2: 0x4C5D 6E7F

### 3.10.4 Flash memory status register (FLASH\_SR)

Address offset: 0x010

Reset value: 0x000X 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PESD	CFGBSY	Res.	BSY
												r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTV ERR	RD ERR	OPTNV	Res.	Res.	Res.	FAST ERR	MISS ERR	PGS ERR	SIZ ERR	PGA ERR	WRP ERR	PROG ERR	Res.	OP ERR	EOP
rc_w1	rc_w1	r				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

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

Bit 19 **PESD**: Programming / erase operation suspended.

This bit is set and reset by hardware.

Set when at least one PES bit in either *Flash memory access control register (FLASH\_ACR)* or *Flash memory CPU2 access control register (FLASH\_C2ACR)* is set.

Cleared when both PES bits in FLASH\_ACR and FLASH\_C2ACR are cleared.

When set, new program or erase operations are not started.

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

This bit is set and reset by hardware. (set when first word is sent and reset when program operation completes or is interrupted by an error.)

When set to '1' the programming and erase settings in FLASH\_PG and FLASH\_PNB requested by *Flash memory control register (FLASH\_CR)* are used (busy), and cannot be changed (a programming or erase operation is ongoing).

When reset to '0' programming and erase settings in FLASH\_PG and FLASH\_PNB in *Flash memory control register (FLASH\_CR)* can be modified.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **BSY**: Busy

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

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

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

Cleared by writing 1.

Bit 14 **RDERR**: PCROP read error

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

Cleared by writing 1.

Bit 13 **OPTNV**: User option OPTVAL indication.

This bit is set and reset by hardware.

0: The OBL user option OPTVAL indicates "valid".

1: The OBL user option OPTVAL indicates "not valid".

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **FASTERR**: Fast programming error

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

Cleared by writing 1.

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

In fast programming mode, 64 double words (512 bytes) must be sent to flash memory successively, and the new data must be sent to the logic control before the current data is fully programmed. MISSERR is set by hardware when the new data is not present in time.

Cleared by writing 1.

Bit 7 **PGSERR**: Programming sequence error

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

Cleared by writing 1.

Bit 6 **SIZERR**: Size error

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

Cleared by writing 1.

Bit 5 **PGAERR**: Programming alignment error

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

Cleared by writing 1.

Bit 4 **WRPERR**: Write protection error

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

Cleared by writing 1.

Bit 3 **PROGERR**: Programming error

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

Cleared by writing 1.

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

Bit 1 **OPERR**: Operation error

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

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

Cleared by writing '1'.

Bit 0 **EOP**: End of operation

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

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

Cleared by writing 1.

### 3.10.5 Flash memory control register (FLASH\_CR)

Address offset: 0x014

Reset value: 0xC000 0000

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

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

- When the PESD bit in *Flash memory status register (FLASH\_SR)* is cleared, the register write access is stalled until the CFGBSY bit is cleared. (by the other CPU)
- When the PESD bit in *Flash memory status register (FLASH\_SR)* is set and a program or an erase operation is ongoing, the register write access causes a bus error.
- When the PESD bit in *Flash memory status register (FLASH\_SR)* is set, but there is no ongoing programming or erase operation, the register write access is completed. The requested program or erase operation is suspended, the BSY/CFGBSY asserted and remains 1, until suspend is deactivated. Consequently, PESD bit goes back to 0 and the suspended operation completes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPT LOCK	Res.	Res.	OBL_LAUNCH	RD_ERRIE	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPT STRT	STRT
rs	rs			rc_w1	rw	rw	rw						rw	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.				PNB[7:0]					MER	PER	PG
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

#### Bit 31 **LOCK**: FLASH\_CR lock

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

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

#### Bit 30 **OPTLOCK**: Options lock

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

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

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

#### Bit 27 **OBL\_LAUNCH**: Forces the option byte loading

When set to 1, this bit forces the option byte reloading. This bit is cleared only when the option byte loading is complete. It cannot be written if OPTLOCK is set.

- 0: Option byte loading complete
- 1: Option byte loading requested

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

This bit enables the interrupt generation when the RDERR bit in the FLASH\_SR is set to 1.

- 0: PCROP read error interrupt disabled
- 1: PCROP read error interrupt enabled

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

This bit enables the interrupt generation when the OPERR bit in the FLASH\_SR is set to 1.

- 0: OPERR error interrupt disabled
- 1: OPERR error interrupt enabled

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

This bit enables the interrupt generation when the EOP bit in the FLASH\_SR is set to 1.

- 0: EOP Interrupt disabled
- 1: EOP Interrupt enabled

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

Bit 18 **FSTPG**: Fast programming

- 0: Fast programming disabled
- 1: Fast programming enabled

Bit 17 **OPTSTRT**: Options modification start

This bit triggers an options operation when set.

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

Bit 16 **STRT**: Start

This bit triggers an erase operation when set. If MER and PER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition must be forbidden.

This bit is set only by software, and is cleared when the BSY bit is cleared in FLASH\_SR. Starting operations by the CPU1, involving secure flash memory pages are rejected and a bus error is generated.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:3 **PNB[7:0]**: Page number selection

These bits select the page to erase:

- 0x00: page 0
- 0x01: page 1
- ...
- 0x7F: page 127
- ... (STM32WB55xx only)
- 0xFF: page 255 (STM32WB55xx only)

Bit 2 **MER**: Mass erase

This bit triggers the mass erase (all user pages) when set.

Bit 1 **PER**: Page erase

- 0: page erase disabled
- 1: page erase enabled

Bit 0 **PG**: Programming

- 0: Flash memory programming disabled
- 1: Flash memory programming enabled

### 3.10.6 Flash memory ECC register (FLASH\_ECCR)

Address offset: 0x018

Reset value: 0x0000 0000

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

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

Bit 31 **ECCD**: ECC detection

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

Cleared by writing 1.

Bit 30 **ECCC**: ECC correction

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

Cleared by writing 1.

Bit 29 Reserved, must be kept at reset value.

Bits 28:26 **CPUID[2:0]**: CPU identification

Set by hardware, indicates the Bus-ID of the CPU access causing the ECC failure.

000: CPU1 access

001: CPU2 access

Bit 25 Reserved, must be kept at reset value.

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

0: ECCC interrupt disabled

1: ECCC interrupt enabled

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

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

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

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:0 **ADDR\_ECC[16:0]**: ECC fail double-word address

These bits indicate that double word address is concerned by the ECC error correction or causes the double ECC error detection.

### 3.10.7 Flash memory option register (FLASH\_OPTR)

Address offset: 0x020

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxx1 xxxx xxxx (the option bits are loaded with values from flash memory at reset release)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
			AGC_TRIM[2:0]	Res.	n BOOT0	nSW BOOT0	SRAM2 _RST	SRAM2 _PE	n BOOT1	Res.	Res.	Res.	WWDG _SW	IWDG _STDBY	IWDG _STOP	IWDG _SW
rw	rw	rw		rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	nRST _SHDW	nRST _STDBY	nRST _STOP		BOR_lev[2:0]		ESE						RDP[7:0]			
	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **AGC\_TRIM[2:0]**: Radio automatic gain control trimming

Default value 0b001

Bit 28 Reserved, must be kept at reset value.

Bit 27 **nBOOT0**: nBOOT0 option bit

If nSWBOOT0 bit configuration select BOOT0 is taken from bit nBOOT0, together with bit nBOOT1, selects boot from the user flash memory, SRAM1, or system flash memory. Refer to [Section 2.3: Boot configuration](#).

0: nBOOT0 = 0

1: nBOOT0 = 1

Bit 26 **nSWBOOT0**: Software BOOT0 selection

0: BOOT0 taken from the option bit nBOOT0

1: BOOT0 taken from PH3/BOOT0 pin

Bit 25 **SRAM2\_RST**: SRAM2 and PKA RAM Erase when system reset

0: SRAM2 and PKA RAM erased when a system reset occurs

1: SRAM2 and non-secure PKA RAM not erased when a system reset occurs

Bit 24 **SRAM2\_PE**: SRAM2 parity check enable

0: SRAM2 parity check enabled

1: SRAM2 parity check disabled

Bit 23 **nBOOT1**: Boot configuration

Together with the BOOT0 pin or option bit nBOOT0 (depending on nSWBOOT0 option bit configuration), this bit selects boot mode from the user flash memory, SRAM1 or the System memory. Refer to [Section 2.3: Boot configuration](#).

Bits 22:20 Reserved, must be kept at reset value.

Bit 19 **WWDG\_SW**: Window watchdog selection

0: Hardware window watchdog

1: Software window watchdog

Bit 18 **IWDG\_STDBY**: Independent watchdog counter freeze in Standby mode

0: Independent watchdog counter is frozen in Standby mode

1: Independent watchdog counter is running in Standby mode

Bit 17 **IWDG\_STOP**: Independent watchdog counter freeze in Stop mode

0: Independent watchdog counter is frozen in Stop mode

1: Independent watchdog counter is running in Stop mode

Bit 16 **IWDG\_SW**: Independent watchdog selection

0: Hardware independent watchdog

1: Software independent watchdog

Bit 15 Reserved, must be kept at reset value.

Bit 14 **nRST\_SHDW:**

- 0: Reset generated when entering the Shutdown mode
- 1: No reset generated when entering the Shutdown mode

Bit 13 **nRST\_STDBY:**

- 0: Reset generated when entering the Standby mode
- 1: No reset generated when entering the Standby mode

Bit 12 **nRST\_STOP:**

- 0: Reset generated when entering the Stop mode
- 1: No reset generated when entering the Stop mode

Bits 11:9 **BORLEV[2:0]: BOR reset level**

These bits contain the  $V_{DD}$  supply level threshold that activates/releases the reset.

- 000: BOR Level 0. Reset level threshold is  $\sim 1.7$  V
- 001: BOR Level 1. Reset level threshold is  $\sim 2.0$  V
- 010: BOR Level 2. Reset level threshold is  $\sim 2.2$  V
- 011: BOR Level 3. Reset level threshold is  $\sim 2.5$  V
- 100: BOR Level 4. Reset level threshold is  $\sim 2.8$  V

Bit 8 **ESE: System security enabled flag.**

Indicates whether the system security is enabled user flash memory FSD = 0.

- 0: Security disabled
- 1: Security enabled

Bits 7:0 **RDP[7:0]: Read protection level**

- 0xAA: Level 0, read protection not active
- 0xCC: Level 2, chip read protection active
- Others: Level 1, memories read protection active

*Note: Take care about PCROP\_RDP configuration in Level 1. Refer to [Level 1: Read protection](#) for more details.*

### 3.10.8 Flash memory PCROP zone A start address register (FLASH\_PCROP1ASR)

Address offset: 0x024

Reset value: 0xFFFFFFFF

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PCROP1A_STRT[8:0]														
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 8:0 **PCROP1A\_STRT[8:0]: PCROP1A area start offset**

PCROP1A\_STRT contains the first 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB35xx devices.

### 3.10.9 Flash memory PCROP zone A end address register (FLASH\_PCROP1AER)

Address offset: 0x028

Reset value: 0xXXXX XXXX

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCROP_RDP	Res.														
rs															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	rw						
PCROP1A_END[8:0]															
									rw						

Bit 31 **PCROP\_RDP**: PCROP area preserved when RDP level decreased

This bit is set only. It is reset after a full mass erase due to a change of RDP from Level 1 to Level 0.

0: PCROP area is not erased when the RDP level is decreased from Level 1 to Level 0.

1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:9 Reserved, must be kept at reset value.

Bits 8:0 **PCROP1A\_END[8:0]**: PCROP1A area end offset

PCROP1A\_END contains the last 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB35xx devices.

### 3.10.10 Flash memory WRP area A address register (FLASH\_WRP1AR)

Address offset: 0x02C

Reset value: 0xXXXX XXXX

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

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **WRP1A\_END[7:0]**: WRP first area “A” end offset

Contains the last 4 Kbytes page of the WRP first area. Note that bit 23 is reserved on STM32WB35xx devices.

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

Bits 7:0 **WRP1A\_STRT[7:0]**: WRP first area “A” start offset

Contains the first 4 Kbytes page of the WRP first area. Note that bit 7 is reserved on STM32WB35xx devices.

### 3.10.11 Flash memory WRP area B address register (FLASH\_WRP1BR)

Address offset: 0x030

Reset value: 0xXXXX XXXX

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

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **WRP1B\_END[7:0]**: WRP second area “B” end offset

WRP1B\_END contains the last 4 Kbytes page of the WRP second area. Note that bit 23 is reserved on STM32WB35xx devices.

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

Bits 7:0 **WRP1B\_STRT[7:0]**: WRP second area “B” start offset

WRP1B\_END contains the first 4 Kbytes page of the WRP second area. Note that bit 7 is reserved on STM32WB35xx devices.

### 3.10.12 Flash memory PCROP zone B start address register (FLASH\_PCROP1BSR)

Address offset: 0x034

Reset value: 0xXXXX XXXX

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16						
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Res.	PCROP1B_STRT[8:0]																				
								rw	rw	rw	rw	rw	rw	rw	rw						

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

Bits 8:0 **PCROP1B\_STRT[8:0]**: PCROP1B area start offset

Contains the first 2 Kbytes page of the PCROP1B area. Note that bit 8 is reserved on STM32WB35xx devices.

### 3.10.13 Flash memory PCROP zone B end address register (FLASH\_PCROP1BER)

Address offset: 0x038

Reset value: 0xXXXX XXXX

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
PCROP1B_END[8:0]															

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

Bits 8:0 **PCROP1B\_END[8:0]**: PCROP1B area end offset

Contains the last 2 Kbytes page of the PCROP1B area. Note that bit 8 is reserved on STM32WB35xx devices.

### 3.10.14 Flash memory IPCC mailbox data buffer address register (FLASH\_IPCCDBR)

Address offset: 0x03C

Reset value: 0xXXXX XXXX

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

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

Bits 13:0 **IPCCDBA[13:0]**: IPCC mailbox data buffer base address offset

Contains the first double-word offset of the IPCC mailbox data buffer area in SRAM2.

### 3.10.15 Flash memory CPU2 access control register (FLASH\_C2ACR)

Address offset: 0x05C

Reset value: 0x0000 0600

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

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

Bit 15 **PES**: CPU2 program / erase suspend request

0: Flash memory program and erase operations granted.

1: Any new flash memory program and erase operation is suspended until this bit and the same bit in *Flash memory access control register (FLASH\_ACR)* are cleared. The PESD bit in both the *Flash memory status register (FLASH\_SR)* and *Flash memory CPU2 status register (FLASH\_C2SR)* is set when at least one PES bit in FLASH\_ACR or FLASH\_C2ACR is set.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ICRST**: CPU2 Instruction cache reset

0: CPU2 Instruction cache is not reset

1: CPU2 Instruction cache is reset

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

Bit 10 Reserved, must be kept at reset value.

Bit 9 **ICEN**: CPU2 Instruction cache enable

0: CPU2 Instruction cache is disabled

1: CPU2 Instruction cache is enabled

Bit 8 **PRFTEN**: CPU2 Prefetch enable.

0: CPU2 prefetch is disabled

1: CPU2 prefetch is enabled

Bits 7:0 Reserved, must be kept at reset value.

### 3.10.16 Flash memory CPU2 status register (FLASH\_C2SR)

Address offset: 0x060

Reset value: 0x000X 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PESD	CFGBSY	Res.	BSY
												r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RD ERR	Res.	Res.	Res.	Res.	FAST ERR	MISS ERR	PGS ERR	SIZ ERR	PGA ERR	WRP ERR	PROG ERR	Res.	OP ERR	EOP
	rc_w1					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

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

Bit 19 **PESD**: Programming / erase operation suspended.

This bit is set and reset by hardware.

Set when at least one PES bit in either *Flash memory access control register (FLASH\_ACR)* or *Flash memory CPU2 access control register (FLASH\_C2ACR)* is set.

Cleared when both PES bits in FLASH\_ACR and FLASH\_C2ACR are cleared.

When set new program or erase operations are not started.

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

This bit is set and reset by hardware. (set when first word is sent and reset when program operation completes or is interrupt by an error.)

When set to '1' the programming and erase settings in (FLASH\_PG, and FLASH\_PNB in *Flash memory CPU2 control register (FLASH\_C2CR)*) are used (busy) and may not be changed. A flash programming or erase operation is ongoing.

When reset to '0' programming and erase settings in (FLASH\_PG and FLASH\_PNB in *Flash memory CPU2 control register (FLASH\_C2CR)*) may be modified.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **BSY**: Busy

This bit indicates that a flash memory operation requested by *Flash memory CPU2 control register (FLASH\_C2CR)* is in progress. This is set on the beginning of a flash memory operation and reset when the operation finishes or when an error occurs.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RDERR**: PCROP read error

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

Cleared by writing 1.

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

Bit 9 **FASTERR**: Fast programming error

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

Cleared by writing 1.

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

In fast programming mode, 64 double words (512 bytes) must be sent to the flash memory successively, and the new data must be sent to the flash memory logic control before the current data is fully programmed. MISSERR is set by hardware when the new data is not present in time.

Cleared by writing 1.

Bit 7 **PGSERR**: Programming sequence error

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

Cleared by writing 1.

Bit 6 **SIZERR**: Size error

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

Cleared by writing 1.

Bit 5 **PGAERR**: Programming alignment error

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

Cleared by writing 1.

Bit 4 **WRPERR**: Write protection error

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

Cleared by writing 1.

Bit 3 **PROGERR**: Programming error

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

Cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Operation error

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

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

Cleared by writing '1'.

Bit 0 **EOP**: End of operation

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

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

Cleared by writing 1.

### 3.10.17 Flash memory CPU2 control register (FLASH\_C2CR)

Address offset: 0x064

Reset value: 0xC000 0000

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

This register cannot be modified when CFGBSY in *Flash memory CPU2 status register (FLASH\_C2SR)* is set.

- When the PESD bit in *Flash memory CPU2 status register (FLASH\_C2SR)* is cleared, the register write access is stalled until the CFGBSY bit is cleared (by the other CPU).
- When the PESD bit in *Flash memory CPU2 status register (FLASH\_C2SR)* is set, the register write access causes a bus error.
- When the PESD bit in *Flash memory CPU2 status register (FLASH\_C2SR)* is set, but there is no ongoing programming or erase operation, the register write access is completed. The requested program or erase operation is suspended, the BSY/CFGBSY asserted and remains 1 until suspend has been deactivated. Consequently, PESD bit goes back to 0 and the suspended operation completes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RD ERRIE	ERR IE	EOP IE	Res.	Res.	Res.	Res.	Res.	FSTPG	Res.	STRT
					rw	rw	rw						rw		rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	PNB[7:0]								MER	PER	PG
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

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

This bit enables the interrupt generation when the RDERR bit in the FLASH\_SR is set to 1.

0: PCROP read error interrupt disabled

1: PCROP read error interrupt enabled

Bit 25 **ERRIE**: Error interrupt enable

This bit enables the interrupt generation when the OPERR bit in the FLASH\_SR is set to 1.

0: OPERR error interrupt disabled

1: OPERR error interrupt enabled

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

This bit enables the interrupt generation when the EOP bit in the FLASH\_SR is set to 1.

0: EOP Interrupt disabled

1: EOP Interrupt enabled

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

Bit 18 **FSTPG**: Fast programming

0: Fast programming disabled

1: Fast programming enabled

Bit 17 Reserved, must be kept at reset value.

Bit 16 **STRT**: Start

This bit triggers an erase operation when set. If MER and PER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition must be forbidden.

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

Starting operations by the CPU1, involving secure flash memory pages is rejected and a bus error generated.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:3 **PNB[7:0]**: Page number selection

These bits select the page to erase:

0x00: page 0

0x01: page 1

...

0x7F: page 127

... (STM32WB55xx only)

0xFF: page 255 (STM32WB55xx only)

Bit 2 **MER**: Mass erase

This bit triggers the mass erase (all user pages) when set.

Bit 1 **PER**: Page erase

0: Page erase disabled

1: Page erase enabled

Bit 0 **PG**: Programming

0: Flash programming disabled

1: Flash programming enabled

### 3.10.18 Secure flash memory start address register (FLASH\_SFR)

Address offset: 0x080

Reset value: 0bxxxx xxxx xxxx xxxx xxxx1 xxx0 xxxx xxxx

This register provides write access security and can only be written by the CPU2. A write access from the CPU1 is ignored and a bus error generated. On any read access the register value is returned.

Written values are only taken into account after OBL.

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

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **DDS**: Disable CPU2 debug access

0: CPU2 debug access enabled

1: CPU2 debug access disabled

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

Bit 8 **FSD**: Flash memory security disabled.

1: System and flash memory non-secure

0: System and flash memory secure (the secure area of the flash memory is given by SFSA)

Bits 7:0 **SFSA[7:0]**: Secure flash memory start address

SFSA[7:0] contain the start address of the first 4 Kbytes page of the secure flash memory area.

### 3.10.19 Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH\_SRRVR)

Address offset: 0x084

Reset value: 0xXXXX XXXX

This register provides write access security and can only be written by the CPU2. A write access from the CPU1 is ignored and a bus error generated. On any read access the register value is returned.

Written values are only taken into account after OBL.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C2OPT	NBRSD	SNBRSA[4:0]						Res.	BRSD	SBRSA[4:0]					
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
SBRV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- Bit 31 **C2OPT:** CPU2 boot reset vector memory selection.  
0: SBRV offset addresses SRAM1 or SRAM2, from start address 0x2000 0000. SBRV value must be kept within the SRAM area.  
1: SBRV offset addresses flash memory, from start address 0x0800 0000.
- Bit 30 **NBRSD:** Non-backup SRAM2b security disable.  
NBRSD = 1: SRAM2b is non-secure  
NBRSD = 0: SRAM2b is secure. SNRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area.
- Bits 29:25 **SNRSA[4:0]:** Secure non-backup SRAM2b start address  
NBRSD = 0: SRAM2b is secure. SNRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area.
- Bit 24 Reserved, must be kept at reset value.
- Bit 23 **BRSD:** Backup SRAM2a security disable.  
BRSD = 1: SRAM2a is non-secure  
BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.
- Bits 22:18 **SBRSA[4:0]:** Secure backup SRAM2a start address  
BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.
- Bits 17:0 **SBRV[17:0]:** CPU2 boot reset vector  
Contains the word aligned CPU2 boot reset start address offset within the selected memory area by C2OPT.

### 3.10.20 FLASH register map

Table 18. Flash interface register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
0x000	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.									
	Reset value																																												
0x004	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.											
	Reset value																																												
0x008	FLASH_KEYR	KEYR[31:0]																																											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
0x00C	FLASH_OPT_KEYR	OPTKEYR[31:0]																																											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
0x010	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.											
	Reset value																																												
0x014	FLASH_CR	LOCK	OPTLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.												
	Reset value	1	1	0	ECCC	0	CPUID [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										
0x018	FLASH_ECCR	ADDR_ECC[16:0]	SYSF_ECC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
0x020	FLASH_OPTR	AGC_TRIM [2:0]	VWDG_SW	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.										
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
0x024	FLASH_PCROP1ASR	PCROP1A_STRT[8:0]	NRST_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	0	0	0									
	Reset value																																												
0x028	FLASH_PCROP1AER	PCROP1A_END[8:0]	NRST_SHDW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
	Reset value	x	PCROP_RDP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.										
0x02C	FLASH_WRP1AR	WRP1A_END[7:0]	WRP1A_STRT[7:0]	ESE	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							
	Reset value																																												

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

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 4 Radio system

### 4.1 Introduction

The system is ultra low power compliant with the Bluetooth® core specification BLE5.3 and IEEE 802.15.4 standard.

The radio system consists of a 2.4 GHz RF front end and a Bluetooth® Low Energy (BLE) and IEEE 802.15.4 physical layer controller. The system is controlled from the CPU2 that contains the radio lower protocol software layers. Interface to the application running on the CPU2 is provided via mailbox message system.

### 4.2 Main features

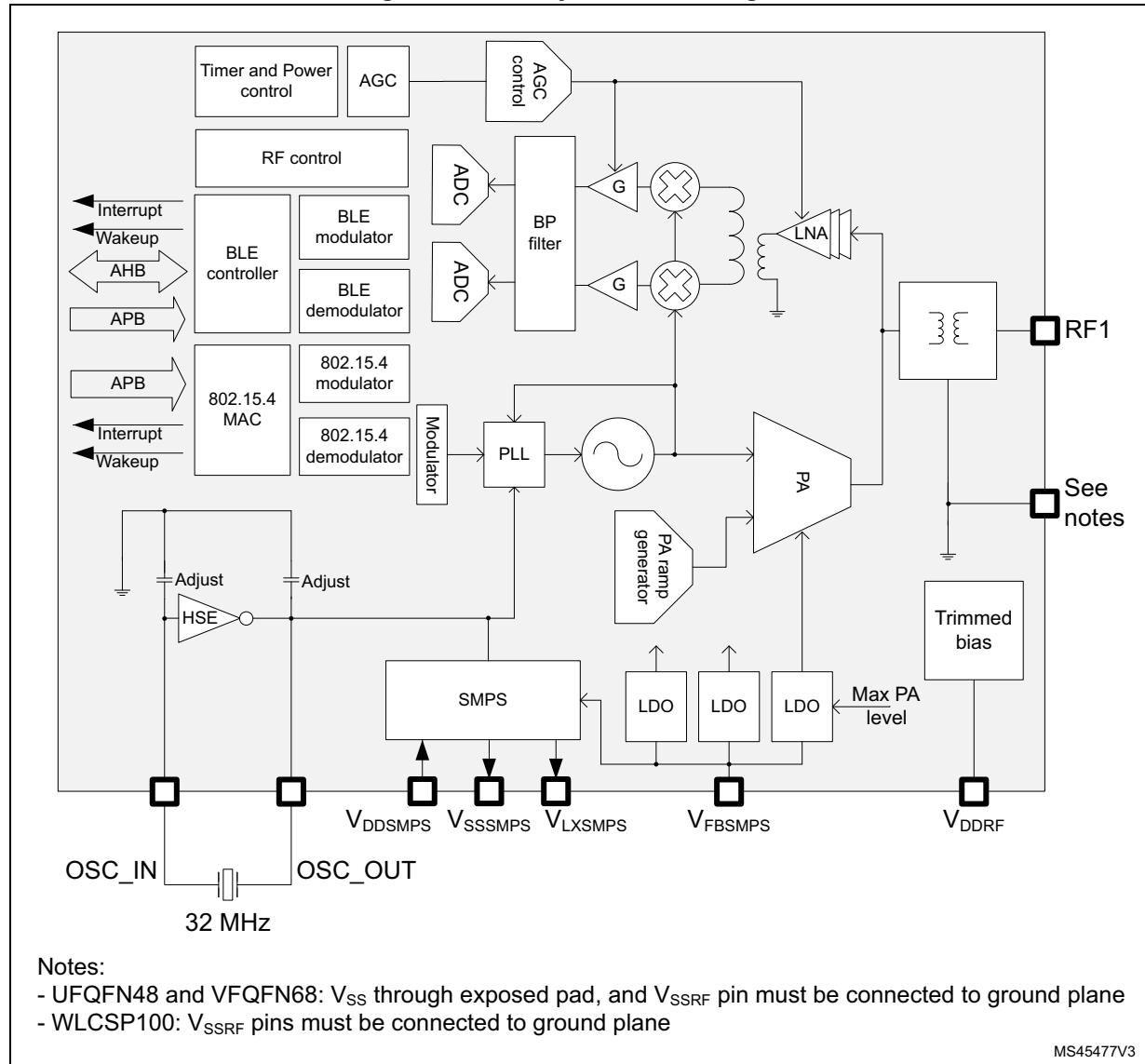
- 2.4 GHz RF transceiver supporting:
  - Bluetooth® BLE5.3 (2 Mbps) standard support
  - IEEE 802.15.4-2011 standard support
- Programmable output power
- RSSI
- Integrated balun
- Bluetooth® BLE5.3 features:
  - GAP: central, peripheral, observer and broadcaster roles
  - Simultaneous multiple role support
  - Master / slave support
  - ATT / GATT: client and server

## 4.3 Radio system functional description

### 4.3.1 General description

The block diagram of the Radio system is shown in [Figure 5](#).

**Figure 5. Radio system block diagram**



The maximum default transmit output power can be supported with a default  $V_{FBSMPS}$  supply level. For higher output power the  $V_{FBSMPS}$  supply level must be increased (see the device data sheet for more information).

MS45477V3

## 5 Cyclic redundancy check calculation unit (CRC)

### 5.1 Introduction

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

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

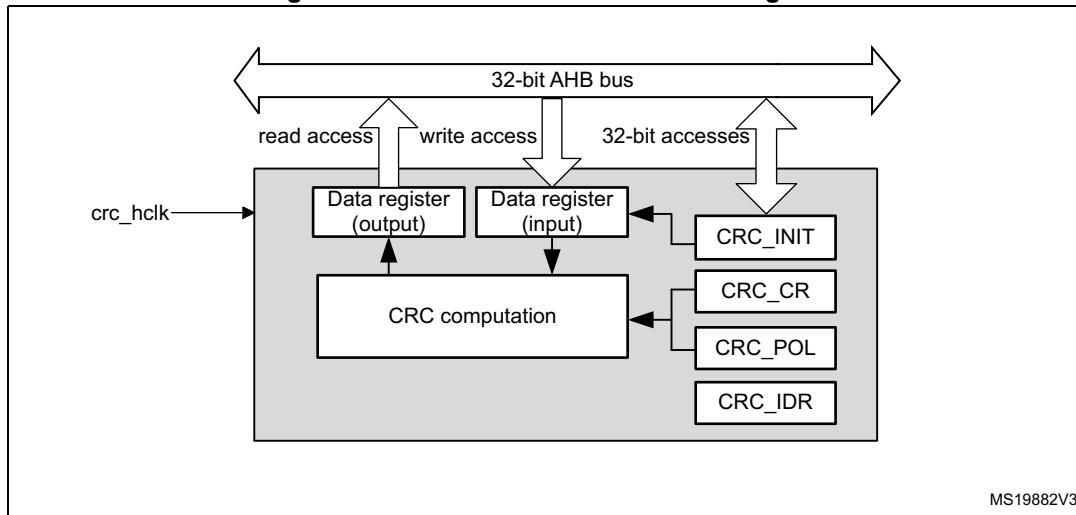
### 5.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-, 16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC\_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

## 5.3 CRC functional description

### 5.3.1 CRC block diagram

Figure 6. CRC calculation unit block diagram



### 5.3.2 CRC internal signals

Table 19. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

### 5.3.3 CRC operation

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

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

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

The duration of the computation depends on data width:

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

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

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

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

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

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

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

The operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

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

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

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

## Polynomial programmability

The polynomial coefficients are fully programmable through the CRC\_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC\_CR register. Even polynomials are not supported.

*Note:* The type of an even polynomial is  $X+X^2+\dots+X^n$ , while the type of an odd polynomial is  $1+X+X^2+\dots+X^n$ .

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

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

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

## 5.4 CRC registers

The CRC\_DR register can be accessed by words, right-aligned half-words and right-aligned bytes. For the other registers only 32-bit accesses are allowed.

### 5.4.1 CRC data register (CRC\_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

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

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

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

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

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

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

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

### 5.4.3 CRC control register (CRC\_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

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

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

0: Bit order not affected

1: Bit-reversed output format

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

This bitfield controls the reversal of the bit order of the input data

00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word

11: Bit reversal done by word

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

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

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

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

Address offset: 0x10

Reset value: 0xFFFF FFFF

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

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

This register is used to write the CRC initial value.

#### 5.4.5 CRC polynomial (CRC\_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

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

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

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

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

### 5.4.6 CRC register map

Table 20. CRC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
0x04	CRC_IDR	IDR[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET		
0x10	CRC_INIT	CRC_INIT[31:0]																															
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
0x14	CRC_POL	POL[31:0]																															
		0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	1		

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 6 Power control (PWR)

### 6.1 Power supplies

The devices require a  $V_{DD}$  operating voltage supply between 1.71 V and 3.6 V. Several independent supplies ( $V_{DDSMPS}$ ,  $V_{FBSMPS}$ ,  $V_{DDA}$ ,  $V_{DDRF}$ ,  $V_{DDUSB}$ ,  $V_{LCD}$ ) can be provided for specific peripherals:

- $V_{DD} = 1.71$  V to 3.6 V  
 $V_{DD}$  is the external power supply for the I/Os, the system analog blocks such as reset, power management, internal clocks and low power regulator. It is provided externally through VDD pins.
- $V_{DDSMPS} = 1.71$  V to 3.6 V  
 $V_{DDSMPS}$  is the external power supply for the SMPS step-down converter. It is provided externally through VDDSMPS supply pin, and must be connected to the same supply as  $V_{DD}$ .
- $V_{FBSMPS} = 1.40$  V to 3.6 V  
 $V_{FBSMPS}$  is the external power supply for the main and RF system regulators. It is provided externally through VFBMPS pin, and may be supplied through either the SMPS step-down converter or connected to the same supply as  $V_{DD}$ .
- $V_{DDA} = 1.62$  V (ADCs/COMPs) to 2.4 V (VREFBUF) to 3.6 V  
 $V_{DDA}$  is the external analog power supply for A/D converters, D/A converters, voltage reference buffer, operational amplifiers and comparators. The  $V_{DDA}$  voltage level is independent from the  $V_{DD}$  voltage and must preferably be connected to  $V_{DD}$  when these peripherals are not used. Note that VREFBUF is available only on STM32WB55xx devices.
- $V_{DDRF} = 1.71$  V to 3.6 V  
 $V_{DDRF}$  is the external power supply for the Radio. It is provided externally through the VDDRF pin, and must be connected to the same supply as  $V_{DD}$ .
- $V_{DDUSB} = 3.0$  V to 3.6 V  
 $V_{DDUSB}$  is the external independent power supply for USB transceivers. The  $V_{DDUSB}$  voltage level is independent from the  $V_{DD}$  voltage and must preferably be connected to  $V_{DD}$  when the USB is not used.
- $V_{LCD} = 2.5$  V to 3.6 V (available only on STM32WB55xx devices)  
The LCD controller can be powered either externally through the VLCD pin, or internally from an internal voltage generated by the embedded step-up converter, which can generate a voltage up to 3.6 V if  $V_{DD}$  is higher than 2.0 V.  
VLCD is multiplexed with PC3, which can be used as GPIO when the LCD is not used.
- $V_{BAT} = 1.55$  V to 3.6 V  
 $V_{BAT}$  is the power supply for RTC, external clock 32 kHz oscillator and backup registers (through power switch) when  $V_{DD}$  is not present.

- VREF-, VREF+ (available only on STM32WB55xx devices)

$V_{REF+}$  is the input reference voltage for the ADC. It is also the output of the internal voltage reference buffer when enabled.

- $V_{DDA} < 2$  V:  $V_{REF+}$  must be equal to  $V_{DDA}$
- $V_{DDA} \geq 2$  V:  $V_{REF+}$  must be between 2 V and  $V_{DDA}$

$V_{REF+}$  can be grounded when ADC is not active.

The internal voltage reference buffer supports two output voltages, configured with VRS bit in the VREFBUF\_CSR register:

- $V_{REF+} \approx 2.048$  V: this requires  $V_{DDA}$  equal to or higher than 2.4 V
- $V_{REF+} \approx 2.5$  V: this requires  $V_{DDA}$  equal to or higher than 2.8 V

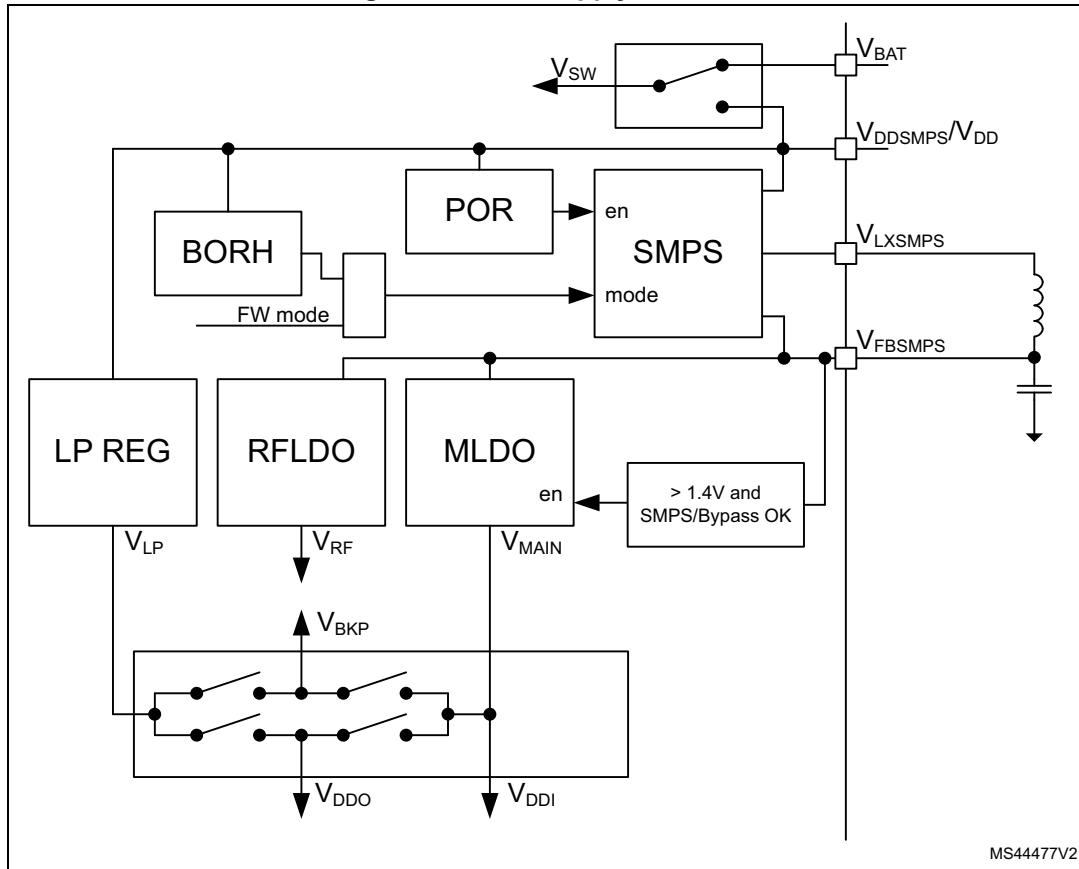
VREF+ pin is not available on all packages, when not available it is internally bonded to VDDA. When the VREF+ is double-bonded with VDDA in a package, the internal voltage reference buffer is not available and must be kept disabled (refer to the datasheet for pinout descriptions).

During power up and power down, the following power sequence is required:

- When  $V_{DD} < 1$  V the other power supplies ( $V_{DDA}$ ,  $V_{DDUSB}$  and  $V_{LCD}$ ) must remain below  $V_{DD} + 0.3$  V. During the power down  $V_{DD}$  can temporarily become lower than the other supplies only if the energy provided to the MCU remains below 1 mJ. This allows the external decoupling capacitors to discharge with different time constants.
- When  $V_{DD} \geq 1$  V all power supplies become independent.

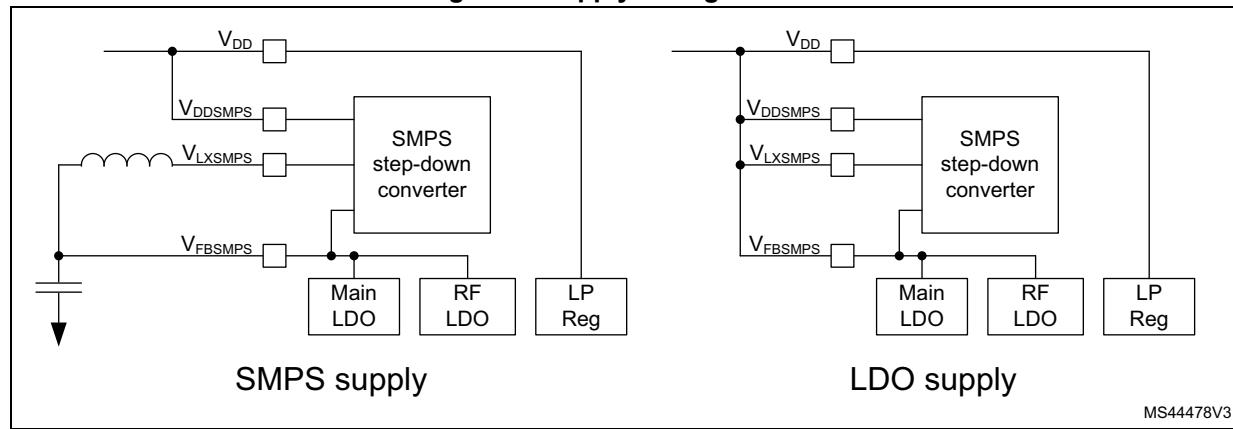
An embedded linear voltage regulator is used to supply the internal digital power  $V_{CORE}$ .  $V_{CORE}$  is the power supply for digital peripherals, SRAM1 and SRAM2. The flash memory is supplied by  $V_{CORE}$  and  $V_{DD}$ .

## Figure 7. Power supply overview



As shown in *Figure 8*, different supply configurations are supported, controlled according to the scheme of *Table 21*.

**Figure 8. Supply configurations**



The SMPS step-down converter operating mode depends upon the SMPSEN setting in [\*PWR control register 5 \(PWR\\_CR5\)\*](#) and upon the system operating modes Run, Stop0, Stop1, Stop2, Standby, and Shutdown.

**Table 21. Supply configuration control**

Supply configuration	SMPSEN	System Operating mode	Description
Default	0	NA	SMPS in Bypass mode
SMPS supply	1 <sup>(1)</sup>	Run	SMPS in SMPS mode
		Stop0	
		Stop1	SMPS in Open mode
		Stop2	
		Standby and Shutdown	
LDO supply	0 <sup>(2)</sup>	Any	SMPS in Bypass mode

1. When SMPS step-down converter SMPS mode is enabled, it is good practice to enable the BORH to monitor the supply.
2. In LDO supply the SMPS must not be enabled, or the product can be permanently damaged.

After a POR reset the SMPS step-down converter is in Bypass mode.

During Stop1, Stop2, and Standby the SMPS step-down converter is put in the Open mode by hardware (see [Table 21](#)). When exiting from low-power modes (except Shutdown) the SMPS step-down converter is set by hardware to the mode selected by the SMPSEN bit (SMPS mode or Bypass). The SMPSEN bit in [PWR control register 5 \(PWR\\_CR5\)](#) is retained in Standby mode.

When the SMPS step-down converter is in SMPS mode the BORH can be configured, with BORHC in [PWR control register 5 \(PWR\\_CR5\)](#), to enable switching on the fly when the supply drops below the SMPS step-down converter SMPS mode operating supply level. When the  $V_{DD}$  supply drops below the selected BORH threshold level the SMPS step-down converter is forced in Bypass mode, register bit SMPSEN is cleared. A SMPSFBF interrupt is generated when enabled. When the  $V_{DD}$  supply rises above the BORH threshold level, a BORHF interrupt is generated when enabled. It is up to the software to switch the SMPS step-down converter back to SMPS mode.

When the SMPS step-down converter is disabled SMPSEN is 0, the SMPS step-down converter is in Bypass mode and the application LDO mode can be used.

### 6.1.1 Independent analog peripherals supply

To improve ADC conversion accuracy and to extend the supply flexibility, the analog peripherals have an independent power supply that can be separately filtered and shielded from noise on the PCB.

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

The  $V_{DDA}$  supply voltage can be different from  $V_{DD}$ . The presence of  $V_{DDA}$  must be checked before enabling any of the analog peripherals supplied by  $V_{DDA}$  (A/D converter, comparators, voltage reference buffer).

The  $V_{DDA}$  supply can be monitored by the Peripheral voltage monitoring, and compared with one threshold (1.65 V for PVM3), refer to [Section 6.2.3](#) for more details.

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

The noise performance of analog signals may be impacted by the integrated SMPS switching. To prevent this the SMPS may be switched off on the fly when measuring analog signals.

### ADC reference voltage (available only on STM32WB55xx)

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

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

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

## 6.1.2 Independent USB transceivers supply

The USB transceivers are supplied from a separate  $V_{DDUSB}$  power supply pin.  $V_{DDUSB}$  range is from 3.0 V to 3.6 V and is completely independent from  $V_{DD}$  or  $V_{DDA}$ .

After reset, the USB features supplied by  $V_{DDUSB}$  are logically and electrically isolated and therefore are not available. The isolation must be removed before using the USB FS peripheral, by setting the USV bit in the [PWR control register 2 \(PWR\\_CR2\)](#), once the  $V_{DDUSB}$  supply is present.

The  $V_{DDUSB}$  supply is monitored by the Peripheral voltage monitoring (PVM1) and compared with the internal reference voltage ( $V_{REFINT} \sim 1.2$  V), refer to [Section 6.2.3](#) for more details.

## 6.1.3 Independent LCD supply (available only on STM32WB55xx)

The VLCD pin is provided to control the contrast of the glass LCD. This pin can be used in two ways:

- it can receive from an external circuitry the desired maximum voltage that is provided on segment and common lines to the glass LCD by the microcontroller
- it can also be used to connect an external capacitor that is used by the microcontroller for its voltage step-up converter. This step-up converter is controlled by software to provide the desired voltage to segment and common lines of the glass LCD.

The voltage provided to segment and common lines defines the contrast of the glass LCD pixels. This contrast can be reduced when the user configures the dead time between frames.

- When an external power supply is provided to the VLCD pin, it must be in the range from 2.5 V to 3.6 V. It does not depend on  $V_{DD}$ .
- When the LCD is based on the internal step-up converter, the VLCD pin should be connected to a capacitor (see the product datasheet for further information).

### 6.1.4 Battery backup domain

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

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

---

**Warning:** During  $t_{RSTTEMPO}$  (temporization at  $V_{DD}$  startup) or after a PDR has been detected, the power switch between  $V_{BAT}$  and  $V_{DD}$  remains connected to  $V_{BAT}$ .  
During the startup phase, if  $V_{DD}$  is established in less than  $t_{RSTTEMPO}$  (refer to the datasheet for its value) 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 recommended to connect an external low-drop diode between this power supply and the  $V_{BAT}$  pin.

---

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

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

- PC13 (STM32WB55xx only), PC14 and PC15, which can be used as GPIO pins
- PC13 (STM32WB55xx only), PC14 and PC15, which can be configured by RTC or LSE (refer to [Section 29.4: RTC functional description](#))
- PA0/RTC\_TAMP2 and PC12/RTC\_TAMP3 (STM32WB55xx only) when configured by the RTC as tamper pins

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

- PC13 (STM32WB55xx only), PC14 and PC15 can be controlled only by RTC or LSE (refer to [Section 29.4: RTC functional description](#))
- PA0/RTC\_TAMP2 and PC12/RTC\_TAMP3 (STM32WB55xx only) when configured by the RTC as tamper pins

#### Backup domain access

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

1. Set the DBP bit in the *PWR control register 1 (PWR\_CR1)* to enable access to the backup domain.

### VBAT battery charging

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

The VBAT charging is done either through a 5 k $\Omega$  resistor or through a 1.5 k $\Omega$  resistor, depending on the VBR<sub>S</sub> bit value in the *PWR control register 4 (PWR\_CR4)*.

The battery charging is enabled by setting VBE bit in the *PWR control register 4 (PWR\_CR4)*, and automatically disabled in VBAT mode.

## 6.1.5 Voltage regulator

Two embedded linear voltage regulators supply all the digital circuitries, except for the Standby circuitry and the backup domain. The main regulator output voltage ( $V_{CORE}$ ) can be programmed by software to two different power ranges (Range 1 and Range 2) to optimize the consumption depending on the system maximum operating frequency (refer to [Section 8.2.10: Clock source frequency versus voltage scaling](#) and to [Section 3.3.4: Read access latency](#)).

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

- In Run, Sleep and Stop0 modes, both regulators are enabled and the main regulator (MR) supplies full power to the  $V_{CORE}$  domain (core, memories and digital peripherals).
- In low-power run and low-power sleep modes, the main regulator is off and the low-power regulator (LPR) supplies low power to the  $V_{CORE}$  domain, preserving the contents of the registers and internal SRAM1 and SRAM2.
- In Stop1 and Stop2 modes the main regulator is off and the low-power regulator (LPR) supplies low power to the  $V_{CORE}$  domain, preserving the contents of the registers and of internal SRAM1 and SRAM2.
- In Standby mode with SRAM2a content preserved (RRS bit is set in the *PWR control register 3 (PWR\_CR3)*), the main regulator (MR) is off and the low-power regulator (LPR) provides the supply to SRAM2a only. The core and digital peripherals (except Standby circuitry and backup domain), SRAM1 and SRAM2b are powered off.
- In Standby mode, both regulators are powered off. The contents of the registers and of SRAM1 and SRAM2 is lost except for the Standby circuitry and the backup domain.
- In Shutdown mode, both regulators are powered off. When exiting from Shutdown mode, a power-on reset is generated. Consequently, the contents of the registers and of both SRAM1 and SRAM2 is lost, except for the backup domain.

## 6.1.6 Dynamic voltage scaling management

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

Dynamic voltage scaling to increase  $V_{CORE}$  is known as “overvolting”, it is used to improve the device performance.

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

- Range 1: High-performance range  
The main regulator provides a typical output voltage at 1.2 V. The system clock frequency can be up to 64 MHz. The Flash memory access time for read access is minimum, write and erase operations are possible.
- Range 2: Low-power range  
The main regulator provides a typical output voltage at 1.0 V. The system clock frequency can be up to 16 MHz. The Flash memory access time for a read access is increased as compared to Range 1; write and erase operations are possible.

Voltage scaling is selected through the VOS bit in the [PWR control register 1 \(PWR\\_CR1\)](#).

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

1. Reduce the system frequency to a value lower or equal to 16 MHz.
2. Adjust number of wait states according new frequency target in Range2 (LATENCY bits in the FLASH\_ACR).
3. Select Range 2 in the VOS bits in the [PWR control register 1 \(PWR\\_CR1\)](#).

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

1. Select Range 1 in the VOS bits in the [PWR control register 1 \(PWR\\_CR1\)](#).
2. Wait until the VOSF flag is cleared in the [PWR status register 2 \(PWR\\_SR2\)](#).
3. Adjust number of wait states according new frequency target in Range1 (LATENCY bits in the FLASH\_ACR).
4. Increase the system frequency.

## 6.2 Power supply supervisor

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

The device has an integrated power-on reset / power-down reset, coupled with a brown-out reset circuitry.

Five BOR thresholds can be selected through option bytes.

The BOR can be used in two operating modes, configured by the BORHC bit in [PWR control register 5 \(PWR\\_CR5\)](#):

- Reset mode, where a system reset is generated when the  $V_{DD}$  supply is below the selected BOR threshold.
- Force SMPS step-down converter in Bypass mode when the  $V_{DD}$  supply is below the selected BOR threshold. When the  $V_{DD}$  supply is below the lowest BOR level a system reset is always generated.

The BOR is active in all power modes except Shutdown mode, and cannot be disabled. The BOR mechanism needs to be enabled, and can be disabled at any time if needed.

### Reset mode

During power-on, the BOR keeps the device under reset until the supply voltage  $V_{DD}$  reaches the specified  $V_{BORx}$  threshold. When  $V_{DD}$  drops below the selected threshold, a device reset is generated. When  $V_{DD}$  is above the  $V_{BORx}$  upper limit, the device reset is released and the system can start.

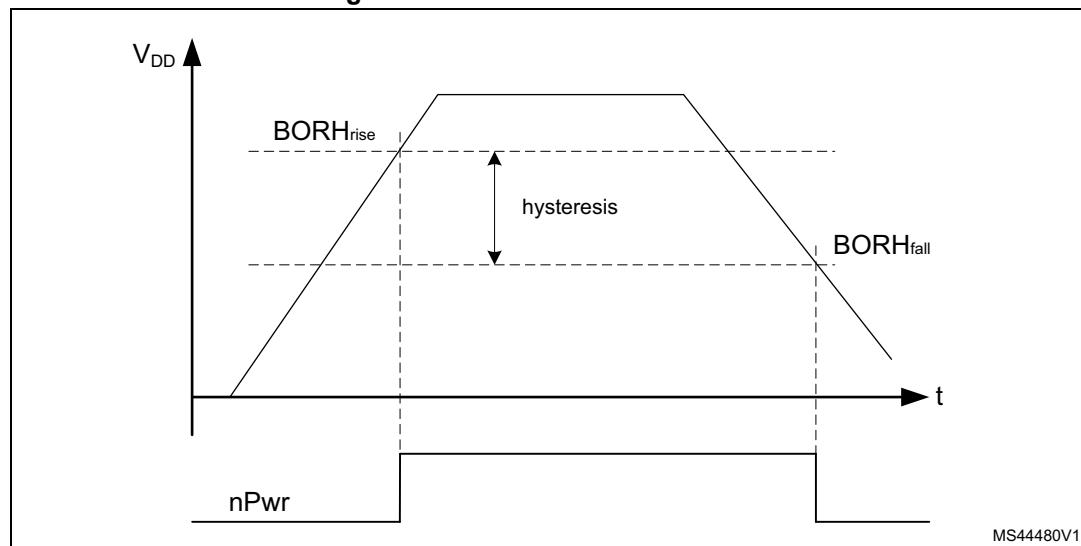
### Force SMPS step-down converter Bypass mode

During power-on, the BOR keeps the device under reset until the supply voltage  $V_{DD}$  reaches the specified  $V_{BOR0}$  threshold. When  $V_{DD}$  drops below this threshold, a device reset is generated. When  $V_{DD}$  is above the  $V_{BOR0}$  upper limit, the device reset is released and the system can start.

To enter and during the SMPS step-down converter in SMPS mode the selected  $BOR_x$  ( $x = 1, 2, 3, 4$ ) threshold is used to determine if SMPS mode is used or if the SMPS step-down converter is forced in Bypass mode. When  $V_{DD}$  drops below the selected threshold, the SMPS step-down converter when entering or in SMPS mode, is forced in Bypass mode. An SMPS step-down converter force Bypass interrupt (SMPSFBF) may be generated when enabled. When  $V_{DD}$  rises above the selected BOR upper limit, a BOR interrupt (BORHF) may be generated when enabled. It is up to software to set the SMPS step-down converter in SMPS mode.

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

Figure 9. Brown-out reset waveform



1. The reset temporization  $t_{RSTTEMPO}$  is present only for the BOR lowest threshold ( $V_{BOR0}$ ).

### 6.2.2 Programmable voltage detector (PVD)

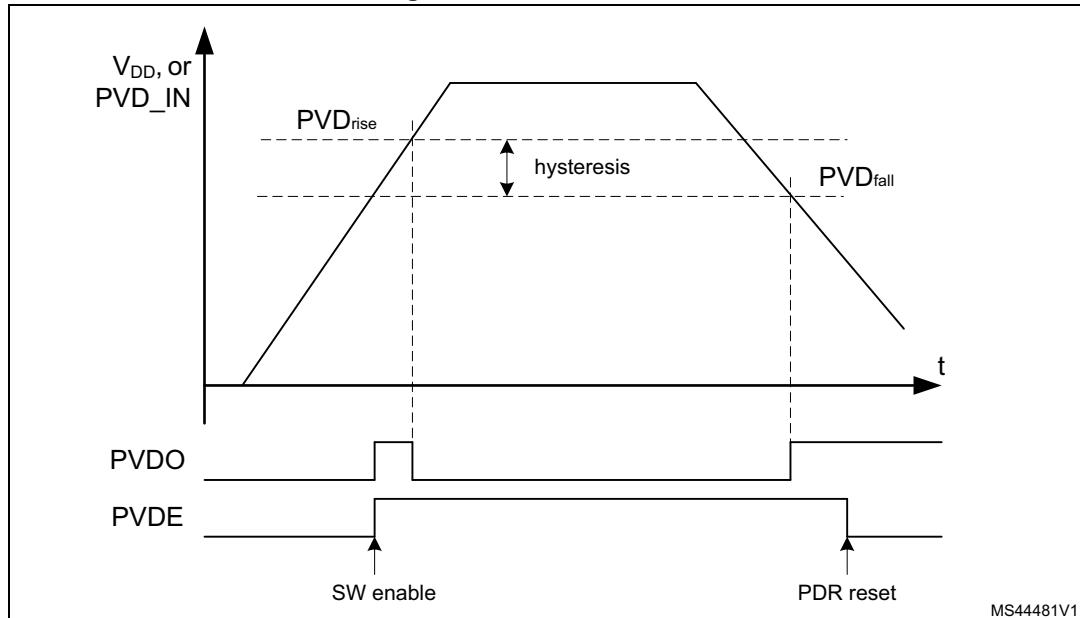
The PVD can be used to monitor the  $V_{DD}$  power supply by comparing it to a threshold selected by the PLS[2:0] bits in the [PWR control register 2 \(PWR\\_CR2\)](#).

The PVD can also be used to monitor a voltage level on the PVD\_IN pin. In this case the voltage level on PVD\_IN is compared to the internal VREFINT level.

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available, in the *PWR status register 2 (PWR\_SR2)*, to indicate if  $V_{DD}$  or the voltage level on PVD\_IN 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  $V_{DD}$  or voltage level on PVD\_IN drops below the PVD threshold and/or when  $V_{DD}$  or voltage level on PVD\_IN rises above the PVD threshold depending on EXTI Line16 rising/falling edge configuration. As an example, the service routine can perform emergency shutdown tasks.

Figure 10. PVD thresholds



### 6.2.3 Peripheral voltage monitoring (PVM)

Only  $V_{DD}$  is monitored by default, as it is the only supply required for all system-related functions. The other supplies ( $V_{DDA}$  and  $V_{DDUSB}$ ) can be independent from  $V_{DD}$  and can be monitored by the peripheral voltage monitoring (PVM).

Each PVM is a comparator between a fixed threshold  $V_{PVMx}$  and the selected power supply.  $PVMOx$  flags indicate if the independent power supply is higher or lower than the  $PVMx$  threshold:  $PVMOx$  flag is cleared when the supply voltage is above the  $PVMx$  threshold, and is set when the supply voltage is below the  $PVMx$  threshold.

Each PVM output is connected to an EXTI line and can generate an interrupt if enabled through the EXTI registers. The  $PVMx$  output interrupt is generated when the independent power supply drops below the  $PVMx$  threshold and/or when it rises above the  $PVMx$  threshold, depending on EXTI line rising/falling edge configuration.

Each PVM can remain active in Stop0, Stop1 and Stop2 modes, and the PVM interrupt can wake up from the Stop mode.

Table 22. PVM features

PVM	Power supply	PVM threshold	EXTI line
PVM1	$V_{DDUSB}$	$V_{PVM1}$ (~ 1.2 V)	31
PVM2	Not used	-	-

**Table 22. PVM features (continued)**

PVM	Power supply	PVM threshold	EXTI line
PVM3	$V_{DDA}$	$V_{PVM3}$ ( $\sim 1.65$ V)	33
PVM4	Not used	-	-

The independent supplies ( $V_{DDA}$  and  $V_{DDUSB}$ ) are not considered as present by default, and a logical and electrical isolation is applied to ignore any information coming from the peripherals supplied by these dedicated sources.

- If these supplies are shorted externally to  $V_{DD}$ , the application must assume that they are available without enabling any PVM.
- If these supplies are independent from  $V_{DD}$ , the PVM can be enabled to confirm whether the supply is present or not.

The following sequence must be applied before using the USB\_FS peripheral:

1. If  $V_{DDUSB}$  is independent from  $V_{DD}$ :
  - a) Enable the PVM1 by setting PVME1 bit in the *PWR control register 2 (PWR\_CR2)*
  - b) Wait for the PVM1 wakeup time
  - c) Wait until PVMO1 bit is cleared in the *PWR status register 2 (PWR\_SR2)*
  - d) Optional: Disable the PVM1 for consumption saving
2. Set the USV bit in the *PWR control register 2 (PWR\_CR2)* to remove the  $V_{DDUSB}$  power isolation.

The following sequence must be applied before using analog to digital converters, digital to analog converters, comparators, operational amplifiers, voltage reference buffer:

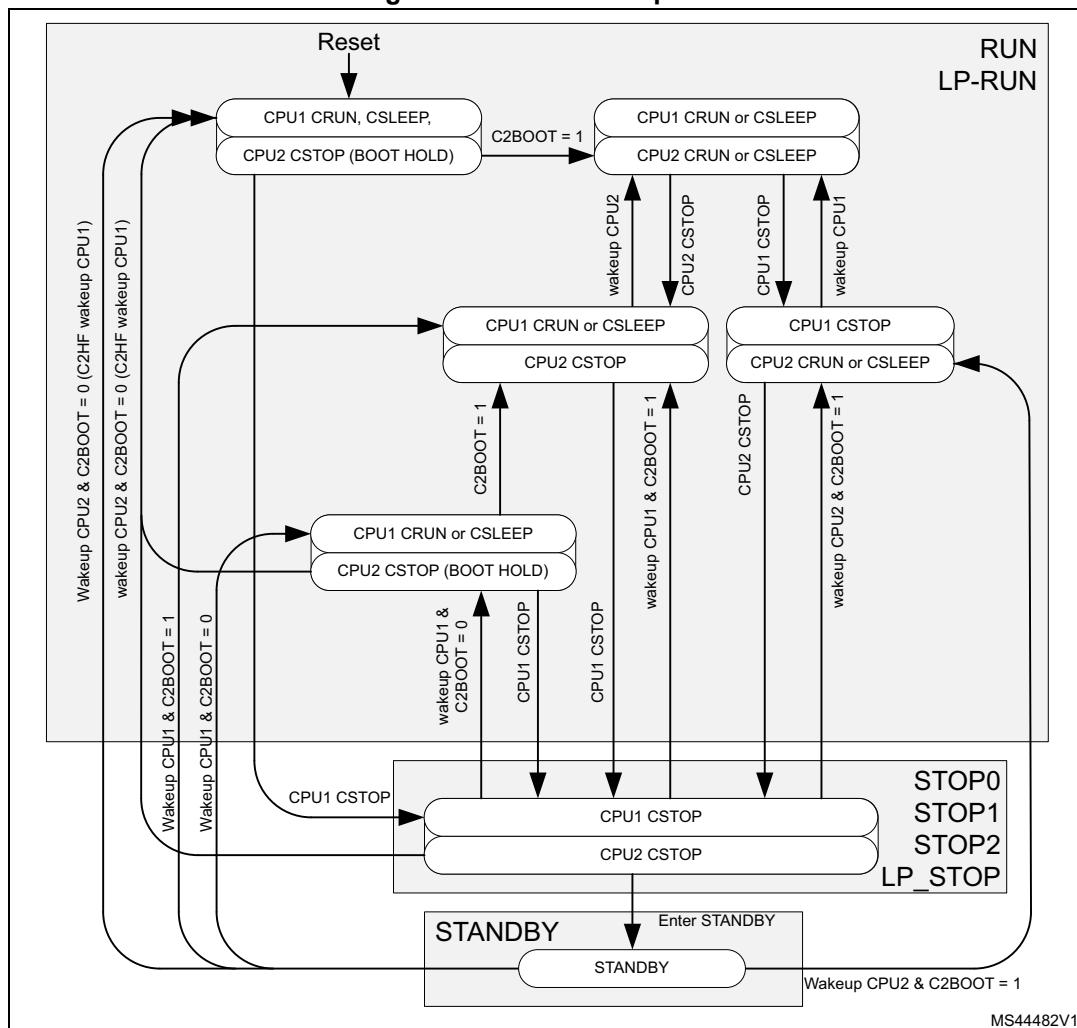
1. If  $V_{DDA}$  is independent from  $V_{DD}$ :
  - a) Enable the PVM3 by setting PVME3 bit in the *PWR control register 2 (PWR\_CR2)*
  - b) Wait for the PVM3 wakeup time
  - c) Wait until PVMO3 bit is cleared in the *PWR status register 2 (PWR\_SR2)*
  - d) Optional: disable the PVM3 for consumption saving
2. Enable the analog peripheral, which automatically removes the  $V_{DDA}$  isolation.

## 6.3 CPU2 boot

Booting of the CPU2 is controlled by the C2BOOT bit in *PWR control register 4 (PWR\_CR4)* register. This allows the CPU1 to initialize the system after a reset or wakeup from system Low-power mode, before booting the CPU2.

- Following reset the CPU2 is prevented from booting by the C2BOOT bit. The CPU2 boots only once the CPU1 has set the C2BOOT bit.
- When exiting from system low power modes STOP0, STOP1, STOP2 or STANDBY the C2BOOT bit controls the CPU2 booting.
  - When C2BOOT = 1, after a system low power mode the CPU2 boots when it is woken up via a wakeup source.
  - When C2BOOT = 0, after a system low power mode the CPU2 is prevented from booting. Instead on a CPU2 wakeup source the CPU1 is woken up via a C2HF. It is up to the CPU1 to boot the CPU2 by setting the C2BOOT bit.

Figure 11. CPU2 boot options



MS44482V1

When the CPU2 is prevented from booting, the wakeup from Low-power mode boot procedure is the following:

- Before the CPU1 enters CSTOP mode it clears the C2BOOT bit.
- When CPU1 exits CSTOP:
  - When the system remains in Run mode, it sets the C2BOOT bit, and subsequently processes the wakeup event.
  - When the system exits from a low power mode from a CPU1 wakeup source, it initializes the system and sets the C2BOOT bit, and subsequently processes the wakeup event.
  - When the system exits from a low power mode from the C2HF wakeup source, it initializes the system and set the C2BOOT bit, and subsequently goes back to CSTOP.

- There is no special wakeup procedure for the CPU2. When the CPU2 wakes up, the system has been initialized by the CPU1. So the CPU2 can directly process the wakeup event.

When the system remains in Run mode (due to the Radio system) the CPU2 wakes up from CSTOP mode independently from the C2BOOT setting.

## 6.4 Low-power modes

By default, the microcontroller is in Run mode after a system or a power Reset and at least one CPU is in CRun mode executing code. Low-power modes are available to save power when the CPU does not need to be kept running, for example when it is waiting for an external event. The user has to select the mode that gives the best compromise between consumption, startup time and available wakeup sources.

The individual CPUs feature two low power modes, entered by the CPU when executing WFI, WFE or on return from an exception handler when SLEEPONEXIT is enabled.

- CSleep mode: when the CPU enters low power mode and SLEEPDEEP is disabled, Arm® “sleep mode”.
- CStop mode: when the CPU enters low power mode and SLEEPDEEP is enabled, Arm® “sleepdeep mode”.

The device features several low-power modes:

- *Sleep mode*: CPU clock off, all peripherals including CPU core peripherals (among them NVIC, SysTick) can run and wake up the CPU when an interrupt or an event occurs.
- *Low-power run mode (LP run)*: This mode is achieved when the system clock frequency is reduced below 2 MHz. The code is executed from the SRAM or from the flash memory. The regulator is in low-power mode to minimize the operating current.
- *Low-power sleep mode (LP sleep)*: This mode is entered from the Low-power run mode: CPU is off.
- *Stop0 mode, Stop1 mode and Stop2 mode*: the content of SRAM1, SRAM2 and of all registers is retained. All clocks in the  $V_{CORE}$  domain are stopped, the PLL, the MSI, the HSI16 and the HSE are disabled. The LSI and the LSE can be kept running.

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

Some peripherals with the wakeup capability can enable the HSI16 RC during Stop mode to detect their wakeup condition.

In Stop2 mode, most of the  $V_{CORE}$  domain is put in a lower leakage mode. Stop1 offers the largest number of active peripherals and wakeup sources, a smaller wakeup time but a higher consumption compared with Stop2. In Stop0 mode, the main regulator remains ON, resulting in the fastest wakeup time but with much higher consumption. The active peripherals and wakeup sources are the same as in Stop1 mode.

The system clock, when exiting from Stop0, Stop1 or Stop2 mode, can be either MSI up to 48 MHz or HSI16, depending on the software configuration.

- *Standby mode*:  $V_{CORE}$  domain is powered off. However, it is possible to preserve the SRAM2a contents:
  - Standby mode with SRAM2a retention when bit RRS is set in the *PWR control register 3 (PWR\_CR3)*. In this case, SRAM2a is supplied by the low-power regulator.
  - Standby mode when bit RRS is cleared in the *PWR control register 3 (PWR\_CR3)*.

In this case the main regulator and the low-power regulator are powered off.

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

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

The system clock, when exiting Standby modes, is HSI16.

- **Shutdown mode:**  $V_{CORE}$  domain is powered off. All clocks in the  $V_{CORE}$  domain are stopped, the PLL, the MSI, the HSI16, the LSI and the HSE are disabled. The LSE can be kept running. The system clock, when exiting Shutdown mode, is MSI at 4 MHz. In this mode, the supply voltage monitoring is disabled and the product behavior is not guaranteed in case of a power voltage drop.

**Note:** *Stop, Standby, and Shutdown modes are only entered when both CPUs are in CStop mode.*

In addition, the power consumption in Run mode can be reduced by slowing down the system clocks, and/or by gating the clocks to the APB and AHB peripherals when they are unused.

The system operation mode depend on the CPU1, the CPU2 and the Radio sub-system operating mode. The system enters a low power mode only when all three sub-systems allow it to do so.

After a system reset the CPU1 is in CRUN mode. The CPU2 boots only if enabled by the CPU1 via the C2BOOT register bit. As long as the CPU1 does not boot the CPU2, the device operates as a single CPU system. The CPU1 can enter and wakeup from system low power modes on its own.

When CPU2 has boot, the CPU1, CPU2 and Radio sub-systems can enter and wakeup from system low power modes on their own. The different wakeup sources for the different sub-systems are detailed in [Table 23](#).

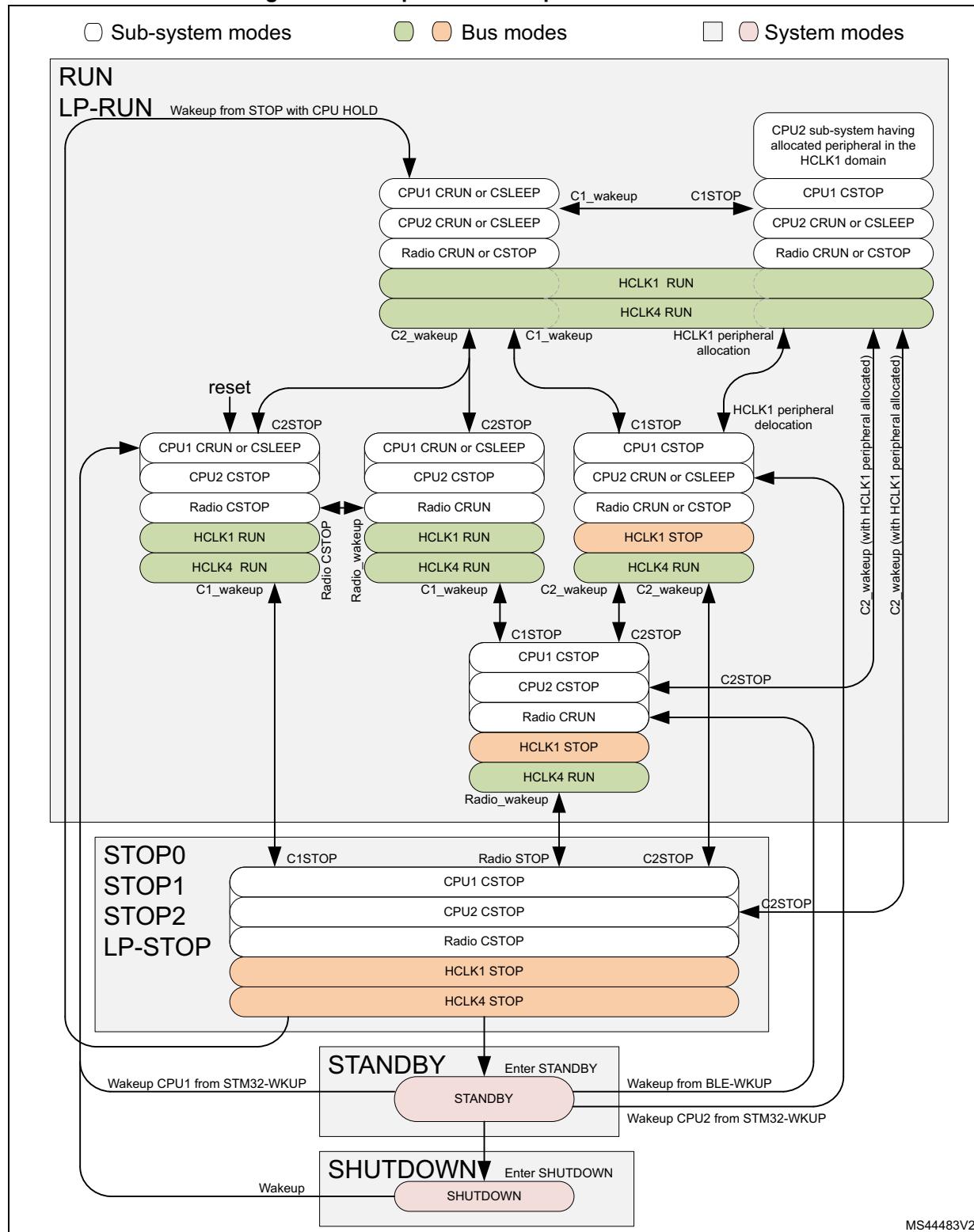
**Table 23. Sub-system low power wakeup sources**

Wakeup source	CPU1	CPU2	Radio
EXTI	From Stop modes	From Stop modes	Not available
RTC	From Stop and Standby modes	From Stop and Standby modes	Not available
WKUP	From Stop and Standby modes	From Stop and Standby modes	Not available
SMPS	From Stop	From Stop	Not available
RADIO	From Stop	From Stop	Not available
RFWAKEUP	Not available	From Stop and Standby modes	From Stop and Standby modes

The system low power mode to enter depends on the allowed mode selected by both CPUs in the LPMS bits of [PWR control register 1 \(PWR\\_CR1\)](#) and [PWR CPU2 control register 1 \(PWR\\_C2CR1\)](#). This is also valid when CPU2 is kept in hold by C2BOOT.

[Figure 12](#) shows the operating modes state diagram. The CPU1, CPU2 and Radio sub-systems operate interdependently according to their own sub-system states. Each sub-system has its own wakeup sources, which allow it to wakeup from Stop and Standby modes. For the device to be in Stop, Standby or Shutdown mode, all three sub-systems need to be in CStop. When one sub-system enters CRun mode the device enters Run mode.

Figure 12. Low-power modes possible transitions



MS44483V2

Table 24. Low-power mode summary

Mode name	Entry	Wakeup source <sup>(1)</sup>	Wakeup system clock	Effect on clocks	Voltage regulator			
					MR	LPR		
Sleep (Sleep-now or Sleep-on-exit)	WFI or Return from ISR	Any interrupt	Same as before entering Sleep mode	CPU clock OFF No effect on other clocks or analog clock sources	ON			
	WFE	Wakeup event						
Low-power run	Set LPR bit	Clear LPR bit	Same as Low-power run clock	None	OFF	ON		
Low-power sleep	Set LPR bit + WFI or Return from ISR	Any interrupt	Same as before entering Low-power sleep mode	CPU clock OFF No effect on other clocks or analog clock sources				
	Set LPR bit + WFE	Wakeup event						
Stop0	LPMS = "000" + SLEEPDEEP bit + WFI or Return from ISR or WFE	Any EXTI line (configured in the EXTI registers). Specific peripherals events	HSI16 when STOPWUCK = 1 in RCC_CFRG. MSI with the frequency before entering Stop mode when STOPWUCK = 0.	All clocks OFF except LSI and LSE	ON			
Stop1	LPMS = "001" + SLEEPDEEP bit + WFI or Return from ISR or WFE							
Stop2	LPMS = "010" + SLEEPDEEP bit + WFI or Return from ISR or WFE							
Standby with SRAM2a	LPMS = "011" + Set RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, LSECSS, external reset in NRST pin, IWDG reset	HSI16	All clocks OFF except LSI and LSE	OFF			
Standby	LPMS = "011" + Clear RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, LSECSS, external reset in NRST pin, IWDG reset						
Shutdown	LPMS = "1--" + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, external reset in NRST pin	MSI 4 MHz	All clocks OFF except LSE	OFF	OFF		

1. Refer to [Table 25](#).

Table 25. Functionalities depending on system operating mode<sup>(1)</sup>

Peripheral	Run	Sleep	Low-power run	Low-power sleep	Stop0		Stop1		Stop2		Standby		Shutdown		VBAT
					-	Wakeup capability	-	Wakeup capability							
CPU1	Y	-	Y	-	-	-	-	-	-	-	-	-	-	-	-
CPU2	Y	-	Y	-	-	-	-	-	-	-	-	-	-	-	-
Radio-system (BLE, 802)	Y	Y	-	-	-	Y	-	Y	-	Y	-	Y <sup>(2)</sup>	-	-	-
Flash memory	Y	Y	O	O	R	-	R	-	R	-	R	-	R	-	R
SRAM1	Y	O <sup>(4)</sup>	Y	O <sup>(4)</sup>	R	-	R	-	R	-	-	-	-	-	-
SRAM2a	Y	O <sup>(4)</sup>	Y	O <sup>(4)</sup>	R	-	R	-	R	-	O <sup>(3)</sup>	-	-	-	-
SRAM2b	Y	O <sup>(4)</sup>	Y	O <sup>(4)</sup>	R	-	R	-	R	-	-	-	-	-	-
QUADSPI	O	O	O	O	-	-	-	-	-	-	-	-	-	-	-
Backup registers	Y	Y	Y	Y	R	-	R	-	R	-	R	-	R	-	R
Brown-out reset (BOR)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-
Brown-out SMPS force bypass (BOR)	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-
Programmable voltage detector (PVD)	O	O	O	O	O	O	O	O	O	O	-	-	-	-	-
Peripheral voltage monitor (PVMx; x=1, 3)	O	O	O	O	O	O	O	O	O	O	-	-	-	-	-
DMAx (x=1, 2)	O	O	O	O	-	-	-	-	-	-	-	-	-	-	-
High speed internal (HSI16)	O	O	O	O	O <sup>(5)</sup>	-	O <sup>(5)</sup>	-	-	-	-	-	-	-	-
Oscillator HSI48	O	O	-	-	-	-	-	-	-	-	-	-	-	-	-
High speed external (HSE)	O	O	O	O	-	-	-	-	-	-	-	-	-	-	-
Low speed internal (LSI)	O	O	O	O	O	-	O	-	O	-	O	-	-	-	-
Low speed external (LSE)	O	O	O	O	O	-	O	-	O	-	O	-	O	-	O
Multi-speed internal (MSI)	O	O	O	O	-	-	-	-	-	-	-	-	-	-	-
Clock security system (CSS)	O	O	O	O	-	-	-	-	-	-	-	-	-	-	-
Clock security system on LSE	O	O	O	O	O	O	O	O	O	O	O	O	-	-	-
RTC / Auto wakeup	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
Number of RTC tamper pins	3	3	3	3	3	O	3	O	3	O	3	O	3	O	3
LCD <sup>(6)</sup>	O	O	O	O	O	O	O	O	O	O	-	-	-	-	-
USB FS	O <sup>(7)</sup>	O <sup>(7)</sup>	-	-	-	O	-	O	-	-	-	-	-	-	-
USART1	O	O	O	O	O <sup>(8)</sup>	O <sup>(8)</sup>	O <sup>(8)</sup>	O <sup>(8)</sup>	-	-	-	-	-	-	-
Low-power UART (LPUART)	O	O	O	O	O <sup>(8)</sup>	O <sup>(8)</sup>	O <sup>(8)</sup>	O <sup>(8)</sup>	O <sup>(8)</sup>	O <sup>(8)</sup>	-	-	-	-	-

Table 25. Functionalities depending on system operating mode<sup>(1)</sup> (continued)

Peripheral	Run	Sleep	Low-power run	Low-power sleep	Stop0	Stop1	Stop2	Standby	Shutdown	VBAT
					Wakeup capability					
I2C1	O	O	O	O	O <sup>(9)</sup>	O <sup>(9)</sup>	O <sup>(9)</sup>	O <sup>(9)</sup>	-	-
I2C3	O	O	O	O	O <sup>(9)</sup>	O <sup>(9)</sup>	O <sup>(9)</sup>	O <sup>(9)</sup>	-	-
SPIx (x=1, 2) <sup>(6)</sup>	O	O	O	O	-	-	-	-	-	-
ADC1	O	O	O	O	-	-	-	-	-	-
COMPx (x=1, 2)	O	O	O	O	O	O	O	O	-	-
Temperature sensor	O	O	O	O	-	-	-	-	-	-
Timers (TIMx)	O	O	O	O	-	-	-	-	-	-
Low-power timer 1 (LPTIM1)	O	O	O	O	O	O	O	O	-	-
Low-power timer 2 (LPTIM2)	O	O	O	O	O	O	O	O	-	-
Independent watchdog (IWDG)	O	O	O	O	O	O	O	O	O	-
Window watchdog (WWDG)	O	O	O	O	-	-	-	-	-	-
SysTick timer	O	O	O	O	-	-	-	-	-	-
Touch sensing controller (TSC) <sup>(6)</sup>	O	O	O	O	-	-	-	-	-	-
True random number generator (RNG)	O	O	-	-	-	-	-	-	-	-
AES hardware accelerator	O	O	O	O	-	-	-	-	-	-
CRC calculation unit	O	O	O	O	-	-	-	-	-	-
IPCC	O	-	O	-	-	-	-	-	-	-
HSEM	O	-	O	-	-	-	-	-	-	-
PKA	O	O	O	O	-	-	-	-	-	-
GPIOs	O	O	O	O	O	O	O	O	(10) O <sup>(11)</sup>	(12) O <sup>(11)</sup>

1. Legend: Y = Yes (enabled). O = Optional (disabled by default, can be enabled by software). R = Data retained.  
- = Not available. Gray cells indicate Wakeup capability.
2. SRAM2a content needs to be retained via the PWR\_CR3.RRS bit.
3. The SRAM2a content can optionally be retained when the PWR\_CR3.RRS bit is set,
4. The SRAM clock can be gated on or off.
5. Some peripherals with wakeup from Stop capability can request HSI16 to be enabled. In this case, HSI16 is woken up by the peripheral, and only feeds the peripheral that requested it. HSI16 is automatically put off when the peripheral does not need it anymore.
6. LCD, SPI2 and TSC are available only on STM32WB55xx devices.
7. Voltage scaling Range 1 only.

8. UART and LPUART reception is functional in Stop mode, and generates a wakeup interrupt on Start, address match or received frame event.
9. I2C address detection is functional in Stop mode, and generates a wakeup interrupt in case of address match.
10. I/Os can be configured with internal pull-up, pull-down or floating in Standby mode.
11. The I/Os with wakeup from Standby/Shutdown capability are PA0 and PA2, and (for STM32WB55xx only) PC5, PC12 and PC13.
12. I/Os can be configured with internal pull-up, pull-down or floating in Shutdown mode but the configuration is lost when exiting Shutdown mode.

### Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop0, Stop1, Stop2, Standby or Shutdown mode while the debug features are used. This is because the CPU1 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 41.3.5: Debug and low power modes](#).

## 6.4.1 Run mode

### Slowing down system clocks

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

For more details, refer to [Section 8.4.3: RCC clock configuration register \(RCC\\_CFGR\)](#).

### Peripheral clock gating

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

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

The peripheral clock gating is controlled by the RCC\_AHBxENR and RCC\_APBxENR registers.

Disabling the peripherals clocks in Sleep mode can be performed automatically by resetting the corresponding bit in the RCC\_AHBxSMENR and RCC\_APBxSMENR registers.

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

To further reduce the consumption when the system is in Run mode, the regulator can be configured in low-power mode. In this mode, the system frequency must not exceed 2 MHz. The Radio sub-system cannot be used in low-power Run mode.

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

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

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

### Entering Low-power run mode

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

1. Optional: Jump into the SRAM and power-down the flash memory by setting the FPDR bit in *PWR control register 1 (PWR\_CR1)* and *PWR CPU2 control register 1 (PWR\_C2CR1)*.
2. Decrease the system clock frequency below 2 MHz.
3. Force the regulator in low-power mode by setting the LPR bit in the *PWR control register 1 (PWR\_CR1)*.

Refer to [Table 26](#) on how to enter the Low-power run mode.

### Exiting Low-power run mode

To exit the Low-power run mode, proceed as follows (refer to [Table 26](#)):

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

**Table 26. Low-power run**

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

#### 6.4.3 Entering Low-power mode

Low power modes are entered by the MCU by executing WFI (Wait for Interrupt), or WFE (Wait for Event) instructions, or when the SLEEPONEXIT bit in the CPU1 System control register is set on Return from ISR.

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

#### 6.4.4 Exiting Low-power mode

From Sleep modes, and Stop modes the MCU exit Low-power mode depending on the way the mode was entered:

- If the WFI instruction or Return from ISR was used to enter the Low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter the Low-power mode, the MCU exits the Low-power mode as soon as an event occurs. The wakeup event can be generated either by an NVIC IRQ interrupt, or by an event.
  - In the first case, when SEVONPEND = 0 in the CPU System control register, enabling an interrupt in the peripheral control register and in the NVIC. When the

MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared. Only NVIC interrupts with sufficient priority wakeup and interrupt the MCU.

When SEVONPEND = 1 in the CPU System control register, enabling an interrupt in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared. All NVIC interrupts wake up the MCU, even the disabled ones. Only enabled NVIC interrupts with sufficient priority wake up and interrupt the MCU.

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

From Standby and Shutdown modes the MCU exits Low-power mode through an external reset (NRST pin), an IWDG reset, a rising/falling edge on one of the enabled WKUPx pins, or an RTC event (see [Section 29: Real-time clock \(RTC\)](#)), or a Radio event (for Standby only).

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

The system mode when the CPU wakes up from CStop mode can be determined from the CnSTOPF and CnSBF in [PWR extended status and status clear register \(PWR\\_EXTSCR\)](#).

**Table 27. CPU CSTOP wakeup vs. system operating mode**

System mode	CPU1		CPU2		CPU1 wakeup	CPU2 wakeup
	C1SBF	C1STOPF	C2SBF	C2STOPF		
Run	0	0	0	0	Wakeup from Run	Wakeup from Run
	0	1	0	0	Wakeup from STOP, but system is already in Run due to CPU2	Wakeup from Run
	0	0	0	1	Wakeup from Run	Wakeup from STOP, but system is already in Run due to CPU1
	1	0	0	0	Wakeup from STANDBY, but system is already in Run due to CPU2	Wakeup from Run
	0	0	1	0	Wakeup from Run	Wakeup from STANDBY, but system is already in Run due to CPU1
	1	1	0	0	Wakeup from STANDBY followed by STOP, but system is already in Run due to CPU2	Wakeup from Run
	0	0	1	1	Wakeup from Run	Wakeup from STANDBY followed by STOP, but system is already in Run due to CPU1

**Table 27. CPU CSTOP wakeup vs. system operating mode (continued)**

System mode	CPU1		CPU2		CPU1 wakeup	CPU2 wakeup
	C1SBF	C1STOPF	C2SBF	C2STOPF		
Stop	0	1	0	1	Wakeup from STOP (CPU2 still in CSTOP)	Wakeup from STOP (CPU1 still in CSTOP)
	1	1	0	1	Wakeup from STOP after the system has been in STANDBY (CPU2 still in CSTOP)	Wakeup from STOP (CPU1 still in CSTOP)
	0	1	1	1	Wakeup from STOP (CPU2 is still in CSTOP)	Wakeup from STOP after the system having been in STANDBY (CPU1 still in CSTOP)
Standby	1	0	1	0	Wakeup from STANDBY (CPU2 still in CSTOP)	Wakeup from STANDBY (CPU1 still in CSTOP)
N.A.	Others			Not valid, does not occur		

## 6.4.5 Sleep mode

### I/O states in Sleep mode

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

### Entering Sleep mode

Sleep mode is entered according to [Entering Low-power mode](#), when the SLEEPDEEP bit in the CPU System control register is cleared (see [Table 28](#)).

### Exiting Sleep mode

The MCU exits from Sleep mode (see [Table 28](#)) as indicated in [Exiting Low-power mode](#).

**Table 28. Sleep mode**

Sleep-now mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) is pending Refer to the Cortex® System control register.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt is pending Refer to the Cortex® System control register.

**Table 28. Sleep mode (continued)**

Sleep-now mode	Description
Mode exit	If WFI or return from ISR was used for entry: Interrupt: refer to <a href="#">Table 54: CPU1 vector table</a> and <a href="#">Table 55: CPU2 vector table</a> If WFE was used for entry and SEVONPEND = 0: Wakeup event: refer to <a href="#">Section 13.5: Interrupt list</a> If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to <a href="#">Table 54: CPU1 vector table</a> and <a href="#">Table 55: CPU2 vector table</a> or Wakeup event: refer to <a href="#">Section 13.5: Interrupt list</a>
Wakeup latency	None

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

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

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

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

### Entering Low-power sleep mode

Low-power sleep mode is entered from Low-power run mode as described in [Section 6.4.3](#), when the SLEEPDEEP bit in the Cortex® System control register is cleared.

Refer to [Table 29](#) for details on how to enter the Low-power sleep mode.

### Exiting Low-power sleep mode

The low-power Sleep mode is exited as described in [Section 6.4.4](#). When exiting Low-power sleep mode by issuing an interrupt or an event, the MCU is in Low-power run mode.

Refer to [Table 29](#) for details on how to exit the Low-power sleep mode.

**Table 29. Low-power sleep**

Low-power sleep-now mode	Description
	Low-power sleep mode is entered from the Low-power run mode. WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) is pending Refer to the Cortex® System control register.
Mode entry	Low-power sleep mode is entered from the Low-power run mode. On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt is pending Refer to the Cortex® System control register.

Table 29. Low-power sleep (continued)

Low-power sleep-now mode	Description
Mode exit	If WFI or Return from ISR was used for entry: Interrupt: refer to <a href="#">Table 54: CPU1 vector table</a> and <a href="#">Table 55: CPU2 vector table</a> If WFE was used for entry and SEVONPEND = 0: Wakeup event: refer to <a href="#">Section 13.5: Interrupt list</a> If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to <a href="#">Table 54: CPU1 vector table</a> and <a href="#">Table 55: CPU2 vector table</a> Wakeup event: refer to <a href="#">Section 13.5: Interrupt list</a> After exiting the Low-power sleep mode, the MCU is in Low-power run mode.
Wakeup latency	None

#### 6.4.7 Stop0 mode

The Stop0 mode is based on the CPU deep sleep mode combined with the peripheral clock gating. The voltage regulator is configured in main regulator mode. In Stop0 mode, all clocks in the  $V_{CORE}$  domain are stopped; the PLL, the MSI, the HSI16 and the HSE oscillators are disabled. Some peripherals with wakeup capability (I2Cx (x=1, 3), USART1 and LPUART) can switch on the HSI16 to receive a frame, and switch off the HSI16 after receiving the frame if it is not a wakeup frame. In this case, the HSI16 clock is propagated only to the peripheral requesting it.

SRAM1, SRAM2 and register contents are preserved.

The BOR is always available in Stop0 mode. The consumption increases when thresholds higher than  $V_{BOR0}$  are used.

#### I/O states in Stop0 mode

In the Stop0 mode, all I/O pins keep the same state as in the Run mode.

#### Entering Stop0 mode

The Stop0 mode is entered according to [Section 6.4.3](#), when the SLEEPDEEP bit in the Cortex System control register is set (see [Table 30](#)).

If flash memory programming is ongoing, the Stop0 mode entry is delayed until the operation is completed.

If an access to the APB domain is ongoing, The Stop0 mode entry is delayed until the APB access is finished.

In Stop0 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 30.3: IWDG functional description](#).
- Real-time clock (RTC): this is configured by the RTCEN bit in the [RCC backup domain control register \(RCC\\_BDCR\)](#).
- Internal RC oscillator (LSI): this is configured by the LSIXON bit in the [RCC](#)

*control/status register (RCC\_CSR).*

- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the [RCC backup domain control register \(RCC\\_BDCR\)](#).

Some peripherals can be used in Stop0 mode and add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock, namely LCD, LPTIM1, LPTIM2, I2Cx (x=1, 3), USART1, LPUART.

The comparators as well as the PVMx (x=1, 3) and the PVD can be used in Stop0 mode. If not needed, they must be disabled by software to reduce power consumption.

The ADC,temperature sensor and VREFBUF buffer can consume power during the Stop0 mode, unless they are disabled before entering this mode.

### Exiting Stop0 mode

The Stop0 mode is exited according to what indicated in [Section 6.4.4](#).

Refer to [Table 30](#) for details on how to exit Stop0 mode.

When exiting Stop0 mode by issuing an interrupt or a wakeup event, the HSI16 oscillator is selected as system clock if the bit STOPWUCK is set in [RCC clock configuration register \(RCC\\_CFGR\)](#). The MSI oscillator is selected as system clock if the bit STOPWUCK is cleared. The wakeup time is shorter when HSI16 is selected as wakeup system clock. The MSI selection enables wakeup at higher frequency, up to 48 MHz.

When the voltage regulator operates in low-power mode, an additional startup delay is incurred when waking up from Stop0 mode with HSI16. By keeping the internal regulator ON during Stop0 mode, the consumption is higher but the startup time is reduced.

When exiting Stop0 mode, the MCU is either in Run mode (Range 1 or Range 2 depending on VOS bit in [PWR control register 1 \(PWR\\_CR1\)](#)) or in Low-power run mode if the bit LPR is set in the same register.

**Table 30. Stop0 mode**

Stop0 mode	Description
Mode entry	<p>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex System control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “000” in PWR_CR1 and/or PWR_C2CR1 or higher</li> </ul>
	<p>On Return from ISR while:</p> <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex System control register</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “000” in PWR_CR1 and/or PWR_C2CR1 or higher</li> </ul>
	<p><i>Note: To enter Stop0 mode, all EXTI line pending bits (in <a href="#">EXTI pending register (EXTI_PR1)</a>, and <a href="#">EXTI pending register (EXTI_PR2)</a>), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop0 mode entry procedure is ignored and program execution continues.</i></p>

**Table 30. Stop0 mode**

Stop0 mode	Description
Mode exit	<p>If WFI or Return from ISR was used for entry: Any EXTI line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to <a href="#">Table 54: CPU1 vector table</a> and <a href="#">Table 55: CPU2 vector table</a>.</p> <p>If WFE was used for entry and SEVONPEND = 0: Any EXTI line configured in event mode. Refer to <a href="#">Section 13.5: Interrupt list</a>.</p> <p>If WFE was used for entry and SEVONPEND = 1: Any EXTI line configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to <a href="#">Table 54: CPU1 vector table</a> and <a href="#">Table 55: CPU2 vector table</a>.</p> <p>Wakeup event: refer to <a href="#">Section 13.5: Interrupt list</a></p>
Wakeup latency	Longest wakeup time between: MSI or HSI16 wakeup time and flash memory wakeup time from Stop0 mode.

#### 6.4.8 Stop1 mode

The Stop1 mode is the same as Stop0 mode except that the main regulator is OFF, and only the low-power regulator is ON. Stop1 mode can be entered from Run mode and from Low-power run mode.

Refer to [Table 31](#) for details on how to enter and exit Stop1 mode.

**Table 31. Stop1 mode**

Stop1 mode	Description
Mode entry	<p>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex System control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “001” in PWR_CR1 and/or PWR_C2CR1 or higher</li> </ul> <p>On Return from ISR while:</p> <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex System control register</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “001” in PWR_CR1 and/or PWR_C2CR1 or higher</li> </ul> <p><i>Note: To enter Stop1 mode, all EXTI line pending bits (in EXTI pending register (EXTI_PR1), and EXTI pending register (EXTI_PR2)), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop1 mode entry procedure is ignored and program execution continues.</i></p>

**Table 31. Stop1 mode**

Stop1 mode	Description
Mode exit	If WFI or Return from ISR was used for entry: Any EXTI line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to <a href="#">Table 54: CPU1 vector table</a> and <a href="#">Table 55: CPU2 vector table</a> . If WFE was used for entry and SEVONPEND = 0: Any EXTI line configured in event mode. Refer to <a href="#">Section 13.5: Interrupt list</a> . If WFE was used for entry and SEVONPEND = 1: Any EXTI line configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to <a href="#">Table 54: CPU1 vector table</a> and <a href="#">Table 55: CPU2 vector table</a> . Wakeup event: refer to <a href="#">Section 13.5: Interrupt list</a> .
Wakeup latency	Longest wakeup time between: MSI or HSI16 wakeup time and regulator wakeup time from Low-power mode + flash memory wakeup time from Stop1 mode.

#### 6.4.9 Stop2 mode

The Stop2 mode is based on the CPU deepsleep mode combined with peripheral clock gating. In Stop2 mode, all clocks in the  $V_{CORE}$  domain are stopped, the PLL, the MSI, the HSI16 and the HSE oscillators are disabled. Some peripherals with wakeup capability (I2C3 and LPUART) can switch on the HSI16 to receive a frame, and switch off the HSI16 after receiving the frame if it is not a wakeup frame. In this case the HSI16 clock is propagated only to the peripheral requesting it.

SRAM1, SRAM2 and register contents are preserved.

The BOR is always available in Stop2 mode. The consumption is increased when thresholds higher than  $V_{BOR0}$  are used.

*Note:* *The comparators, LPUART and LPTIM1 outputs are forced to low speed (OSPEEDy = 00) during Stop2 mode.*

##### I/O states in Stop2 mode

In the Stop2 mode, all I/O pins keep the same state as in the Run mode.

##### Entering Stop2 mode

The Stop2 mode is entered as described in [Section 6.4.3](#), when the SLEEPDEEP bit in the Cortex System control register is set (see [Table 32](#)).

Stop2 mode can only be entered from Run mode. It is not possible to enter Stop2 mode from the Low-power run mode.

If flash memory programming is ongoing, the Stop2 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, The Stop2 mode entry is delayed until the APB access is finished.

In Stop2 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 30.3: IWDG functional description](#).
- real-time clock (RTC): this is configured by the RTCEN bit in the [RCC backup domain control register \(RCC\\_BDCR\)](#)
- Internal RC oscillator (LSI): this is configured by the LSIXON bit in the [RCC control/status register \(RCC\\_CSR\)](#).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the [RCC backup domain control register \(RCC\\_BDCR\)](#).

Some peripherals can be used in Stop2 mode and add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock, namely LCD, LPTIM1, I2C3, LPUART.

The comparators, the PVMx (x=1, 3) and the PVD can be used in Stop2 mode. If not needed, they must be disabled by software to reduce power consumption.

The ADC, temperature sensor and VREFBUF<sup>(a)</sup> buffer can consume power during Stop2 mode, unless they are disabled before entering this mode.

All the peripherals that cannot be enabled in Stop2 mode must be either disabled by clearing the Enable bit in the peripheral itself, or put under reset state by setting the corresponding bit in the

- [RCC AHB1 peripheral reset register \(RCC\\_AHB1RSTR\)](#)
- [RCC AHB2 peripheral reset register \(RCC\\_AHB2RSTR\)](#)
- [RCC AHB3 and AHB4 peripheral reset register \(RCC\\_AHB3RSTR\)](#)
- [RCC APB1 peripheral reset register 1 \(RCC\\_APB1RSTR1\)](#)
- [RCC APB1 peripheral reset register 2 \(RCC\\_APB1RSTR2\)](#)
- [RCC APB2 peripheral reset register \(RCC\\_APB2RSTR\)](#)
- [RCC APB3 peripheral reset register \(RCC\\_APB3RSTR\)](#)

### Exiting Stop2 mode

The Stop2 mode is exited according to [Section 6.4.4](#) (see [Table 32](#)).

When exiting Stop2 mode by issuing an interrupt or a wakeup event, the HSI16 oscillator is selected as system clock if the bit STOPWUCK is set in [RCC clock configuration register \(RCC\\_CFGR\)](#). The MSI oscillator is selected as system clock if the bit STOPWUCK is cleared. The wakeup time is shorter when HSI16 is selected as wakeup system clock. The MSI selection allows wakeup at higher frequency, up to 48 MHz.

When exiting Stop2 mode, the MCU is in Run mode (Range 1 or Range 2 depending on VOS bit in PWR\_CR1).

---

a. Available only on Cat. 3 devices.

Table 32. Stop2 mode

Stop2 mode	Description
Mode entry	<p>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex System control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “010” in PWR_CR1 and/or PWR_C2CR1 or higher</li> </ul> <p>On return from ISR while:</p> <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex System control register</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “010” in PWR_CR1 and/or PWR_C2CR1 or higher</li> </ul> <p><i>Note: To enter Stop2 mode, all EXTI line pending bits in EXTI pending register (EXTI_PR1), and EXTI pending register (EXTI_PR2), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop mode entry procedure is ignored and program execution continues.</i></p>
Mode exit	<p>If WFI or Return from ISR was used for entry:</p> <ul style="list-style-type: none"> <li>Any EXTI line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to <a href="#">Table 54: CPU1 vector table</a> and <a href="#">Table 55: CPU2 vector table</a>.</li> </ul> <p>If WFE was used for entry and SEVONPEND = 0:</p> <ul style="list-style-type: none"> <li>Any EXTI line configured in event mode. Refer to <a href="#">Section 13.5: Interrupt list</a>.</li> </ul> <p>If WFE was used for entry and SEVONPEND = 1:</p> <ul style="list-style-type: none"> <li>Any EXTI line configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to <a href="#">Table 54: CPU1 vector table</a> and <a href="#">Table 55: CPU2 vector table</a>.</li> </ul> <p>Any EXTI line configured in event mode. Refer to <a href="#">Section 13.5: Interrupt list</a>.</p>
Wakeup latency	Longest wakeup time between: MSI or HSI16 wakeup time and regulator wakeup time from Low-power mode + flash wakeup time from Stop2 mode.

#### 6.4.10 Standby mode

Standby mode makes it possible to achieve the lowest power consumption with BOR. It is based on the CPU deepsleep mode, with the voltage regulators disabled (except when SRAM2 content is preserved). The PLL, the HSI16, the MSI and the HSE oscillators are also switched off.

SRAM1 and register contents are lost except for registers in the Backup domain and Standby circuitry (see [Figure 7](#)). SRAM2 content can be preserved if the bit RRS is set in the [PWR control register 3 \(PWR\\_CR3\)](#). In this case the Low-power regulator is ON and provides the supply to SRAM2 only.

The BOR is always available in Standby mode. The consumption is increased when thresholds higher than  $V_{BOR0}$  are used.

#### I/O states in Standby mode

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

The RTC outputs on PC13 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. Five wakeup pins (WKUP $x$ ,  $x=1,2\dots 5$ ) and the three RTC tampers are available.

### Entering Standby mode

Standby mode is entered according to [Section 6.4.3](#), when the SLEEPDEEP bit in the Cortex System control register is set.

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

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a reset. See [Section 30.3: IWDG functional description](#).
- Real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC\_BDCR).
- Internal RC oscillator (LSI): this is configured by the LSIXON bit in the Control/status register (RCC\_CSR).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the Backup domain control register (RCC\_BDCR)

### Exiting Standby mode

Standby mode is exited according to [Section 6.4.4](#). The SBF status flag in the [PWR control register 3 \(PWR\\_CR3\)](#) indicates that the MCU was in Standby mode. All registers are reset after wakeup from Standby except for [PWR control register 3 \(PWR\\_CR3\)](#).

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

**Table 33. Standby mode**

Standby mode	Description
	<p>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex System control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “011” in PWR_CR1 and/or PWR_C2CR1 or higher</li> <li>– WUF<math>x</math> bits are cleared in power status register 1 (PWR_SR1)</li> </ul>
Mode entry	<p>On return from ISR while:</p> <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex System control register</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “011” in PWR_CR1 and/or PWR_C2CR1 or higher</li> <li>– WUF<math>x</math> bits are cleared in power status register 1 (PWR_SR1)</li> </ul> <p>The RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup, tamper or timestamp flags) is cleared.</p>
Mode exit	WKUP $x$ pin edge, RTC event, external Reset in NRST pin, IWDG Reset, BOR reset
Wakeup latency	Reset phase

### 6.4.11 Shutdown mode

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

SRAM1, SRAM2 and register contents are lost except for registers in the Backup domain. The BOR is not available in Shutdown mode. No power voltage monitoring is possible in this mode, therefore the switch to Backup domain is not supported.

#### I/O states in Shutdown mode

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

However this configuration is lost when exiting Shutdown mode due to the power-on reset.

The RTC outputs on PC13 are functional in Shutdown mode. PC14 and PC15 used for LSE are also functional. Five wakeup pins (WKUPx,  $x=1,2\dots 5$ ) and the three RTC tampers are available.

#### Entering Shutdown mode

The Shutdown mode is entered according to [Entering Low-power mode](#), when the SLEEPDEEP bit in the Cortex System control register is set.

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

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

- Real-time clock (RTC): this is configured by the RTCEN bit in the [RCC backup domain control register \(RCC\\_BDCR\)](#). Caution: in case of  $V_{DD}$  power-down the RTC content is lost.
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the [RCC backup domain control register \(RCC\\_BDCR\)](#).

#### Exiting Shutdown mode

The Shutdown mode is exited according to [Exiting Low-power mode](#). A power-on reset occurs when exiting from Shutdown mode. All registers (except for the ones in the Backup domain) are reset after wakeup from Shutdown.

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

Table 34. Shutdown mode

Shutdown mode	Description
	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP bit is set in Cortex System control register – No interrupt (for WFI) or event (for WFE) is pending – LPMS = “1XX” in PWR_CR1 and PWR_C2CR1 – WUFx bits are cleared in power status register 1 (PWR_SR1)
Mode entry	On return from ISR while: – SLEEPDEEP bit is set in Cortex System control register – SLEEPONEXT = 1 – No interrupt is pending – LPMS = “1XX” in PWR_CR1 and PWR_C2CR1 – WUFx bits are cleared in power status register 1 (PWR_SR1) The RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup, tamper or timestamp flags) is cleared.
Mode exit	WKUPx pin edge, RTC event, external Reset in NRST pin
Wakeup latency	Reset phase

#### 6.4.12 Auto wakeup from Low-power mode

The RTC can be used to wakeup the MCU from Low-power mode without depending on an external interrupt (Auto-wakeup mode). The RTC provides a programmable time base for waking up from Stop (0, 1 or 2) or Standby mode at regular intervals. For this purpose, two of the three alternative RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the [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 very low-power consumption.
- Low-power internal RC oscillator (LSI)  
This oscillator is designed to add minimum power consumption.

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

- Configure the EXTI Line 18 to be sensitive to rising edge
- Configure the RTC to generate the RTC alarm

To wakeup from Standby mode, there is no need to configure the EXTI Line 18.

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

- Configure the EXTI Line 20 to be sensitive to rising edge
- Configure the RTC to generate the RTC alarm

To wakeup from Standby mode, there is no need to configure the EXTI Line 20.

The LCD Start of frame interrupt can also be used as a periodic wakeup from Stop (0, 1 or 2) mode. The LCD is not available in Standby mode.

The LCD clock is derived from the RTC clock selected by RTCSEL[1:0].

## 6.5 Real-time radio information

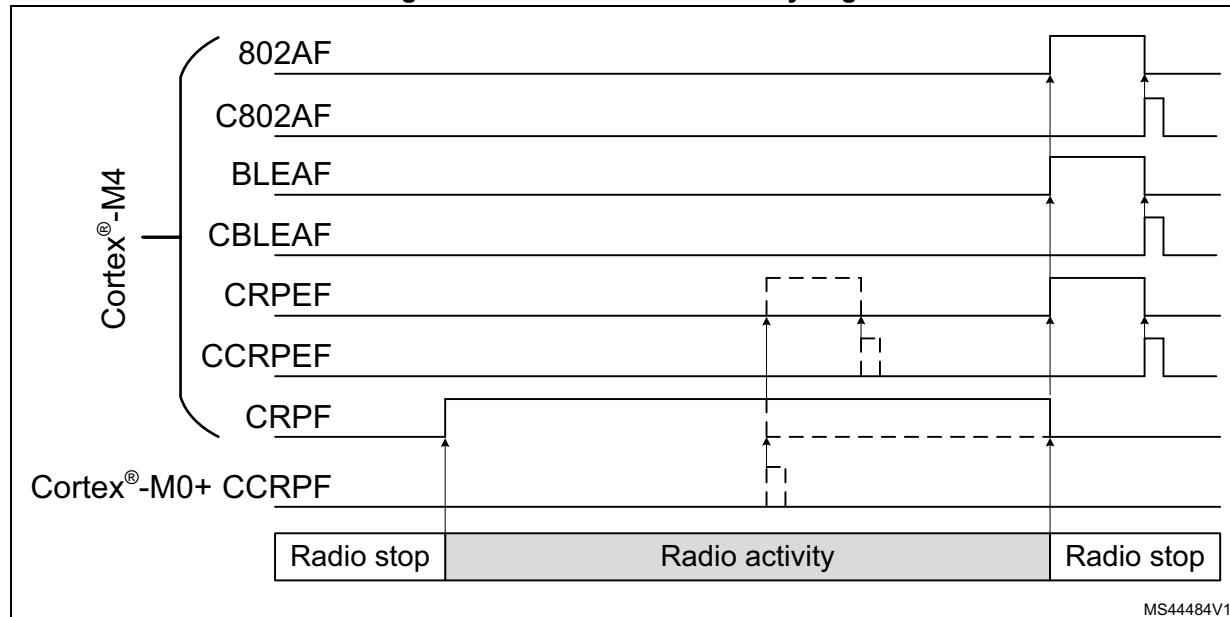
The PWR provides flags indicating the real time operation on the radio:

- IEEE802.15.4 radio activity end interrupt flag
- BLE radio activity end interrupt flag
- Critical radio phase flag
- Critical radio phase end interrupt flag

These flags may be used by the CPU1 to determine the radio activity.

The basic timing relation for the different flags is shown in [Figure 12](#).

**Figure 13. Real-time radio activity flags**



MS44484V1

When enabled, the radio activity end interrupt flag 802AF indicates the end of an IEEE802.15.4 radio activity period, and is generated when the radio enters Cstop mode.

When enabled, the radio activity end interrupt flag BLEAF indicates the end of a BLE radio activity period, and is generated when the radio enters Cstop mode.

The critical radio phase flag indicates the critical real time phase of the radio, where access to the flash memory must not be blocked by erase or program operations (all erase and program operations are suspended by the CPU2 in the flash memory interface). The end of the critical radio phase may be triggered by the CPU2 clearing the critical radio phase flag CRPF, and is eventually cleared at the end of the IEEE802.15.4 or BLE radio activity.

When enabled, the critical radio phase end interrupt flag CRPEF indicates the end of a radio critical phase, and is generated when the radio critical phase ends.

## 6.6 PWR registers

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

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

Address offset: 0x0000 0200

Reset value: 0x0000 0200. This register is reset after wakeup from Standby mode, except for bits [2:0].

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

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

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

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

*Note: Stop2 mode cannot be entered when LPR bit is set. Stop1 is entered instead.*

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

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

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

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

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

- 0: Access to RTC and Backup registers disabled
- 1: Access to RTC and Backup registers enabled

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

Bit 5 **FPDS**: Flash memory power down mode during LPSleep for CPU1

This bit selects whether the flash memory is in power down mode or idle mode when both CPUs are in Sleep mode. flash memory is set in power down mode only when the system is in LPSleep mode and the PWR\_C2CR1.FPDS bit from CPU2 also allows this.

- 0: Flash memory in Idle mode when system is in LPSleep mode
- 1: Flash memory in power down mode when system is in LPSleep mode

Bit 4 **FPDR**: Flash memory power down mode during LPRun for CPU1

This bit can only be written to 1 after unlocking this register bit, by first writing (code 0xC1B0) into this register (when writing the code, the register bits are not updated). Selects whether the flash memory is in power down mode or idle mode when in LPRun mode. (flash memory can only be in power down mode when code is executed from SRAM). Flash memory is set in power down mode only when the system is in LPRun mode, and the PWR\_C2CR1.FPDR bit from CPU2 too allows so.

- 0: Flash memory in Idle mode when system is in LPRun mode
- 1: Flash memory in power down mode when system is in LPRun mode

## Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LPMS[2:0]**: Low-power mode selection for CPU1

These bits are not reset when exiting Standby mode.

These bits select the low-power mode allowed when CPU1 enters the deepsleep mode. The entered system low-power mode depends also upon the PWR\_C2CR1.LPMS allowed low-power mode from CPU2.

- 000: Stop0 mode
- 001: Stop1 mode
- 010: Stop2 mode
- 011: Standby mode
- 1xx: Shutdown mode

*Note: If LPR bit is set, Stop2 mode cannot be selected and Stop1 mode must be entered instead of Stop2.*

*In Standby mode, SRAM2 can be preserved or not, depending on RRS bit configuration in PWR\_CR3.*

## 6.6.2 PWR control register 2 (PWR\_CR2)

Address offset: 0x004

Reset value: 0x0000 0000. This register is reset when exiting Standby mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	USV	Res.	Res.	Res.	PVME3	Res.	PVME1	PLS[2:0]	PLS[2:0]	PLS[2:0]	PVDE
					rw				rw		rw	rw	rw	rw	rw

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

Bit 10 **USV**:  $V_{DDUSB}$  USB supply valid

This bit is used to validate the  $V_{DDUSB}$  supply for electrical and logical isolation purposes. Setting this bit is mandatory to use the USB FS peripheral. If  $V_{DDUSB}$  is not always present in the application, the PVM can be used to determine whether this supply is available or not.

- 0:  $V_{DDUSB}$  is not present. Logical and electrical isolation is applied to ignore this supply.
- 1:  $V_{DDUSB}$  is valid.

Bits 9:7 Reserved, must be kept at reset value.

Bit 6 **PVME3**: Peripheral voltage monitoring 3 enable:  $V_{DDA}$  vs. 1.62 V

- 0: PVM3 ( $V_{DDA}$  monitoring vs. 1.62 V threshold) disabled
- 1: PVM3 ( $V_{DDA}$  monitoring vs. 1.62 V threshold) enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **PVME1**: Peripheral voltage monitoring 1 enable:  $V_{DDUSB}$  vs. 1.2 V

0: PVM1 ( $V_{DDUSB}$  monitoring vs. 1.2 V threshold) disabled.

1: PVM1 ( $V_{DDUSB}$  monitoring vs. 1.2 V threshold) enabled.

Bits 3:1 **PLS[2:0]**: Programmable voltage detector level selection.

These bits select the voltage threshold detected by the Programmable voltage detector:

000:  $V_{PVD0} \sim 2.0$  V

001:  $V_{PVD1} \sim 2.2$  V

010:  $V_{PVD2} \sim 2.4$  V

011:  $V_{PVD3} \sim 2.5$  V

100:  $V_{PVD4} \sim 2.6$  V

101:  $V_{PVD5} \sim 2.8$  V

110:  $V_{PVD6} \sim 2.9$  V

111: External input analog voltage PVD\_IN (compared internally to  $V_{REFINT}$ ). The I/O used as PVD\_IN input must be configured in analog mode in the GPIO register.

*Note: These bits are write-protected when the bit PVDL (PVD lock) is set in the SYSCFG\_CBR register, they are reset only by a system reset.*

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

0: Programmable voltage detector disabled

1: Programmable voltage detector enabled

*Note: This bit is write-protected when bit PVDL (PVD lock) is set in the SYSCFG\_CBR register. This bit is reset only by a system reset.*

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

Address offset: 0x008

Reset value: 0x0000 8000. This register is not reset when exiting Standby modes.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIWUL	EC2H	E802A	EBLEA	ECRPE	APC	RRS	EBORH SMPSFB	Res.	Res.	Res.	EWUP5	EWUP4	EWUP3	EWUP2	EWUP1
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 **EIWUL**: Enable internal wakeup line for CPU1

0: Internal wakeup line interrupt to CPU1 disabled

1: Internal wakeup line interrupt to CPU1 enabled

Bit 14 **EC2H**: Enable CPU2 Hold interrupt for CPU1

Enable CPU2 kept in hold, due to C2BOOT, interrupt to CPU1.

0: Interrupt to CPU1 disabled

1: interrupt to CPU1 enabled

- Bit 13 **E802A**: Enable 802.15.4 end of activity interrupt for CPU1  
 0: Interrupt to CPU1 disabled  
 1: interrupt to CPU1 enabled
- Bit 12 **EBLEA**: Enable BLE end of activity interrupt for CPU1  
 0: Interrupt to CPU1 disabled  
 1: interrupt to CPU1 enabled
- Bit 11 **ECRPE**: Enable critical radio phase end of activity interrupt for CPU1  
 0: Interrupt to CPU1 disabled  
 1: interrupt to CPU1 enabled
- Bit 10 **APC**: Apply pull-up and pull-down configuration from CPU1  
 When this bit for CPU1 or the PWR\_C2CR3.APC bit for CPU2 is set, the I/O pull-up and pull-down configurations defined in the PWR\_PUCRx and PWR\_PDCRx registers are applied. When both bits are cleared, the PWR\_PUCRx and PWR\_PDCRx registers are not applied to the I/Os.
- Bit 9 **RRS**: SRAM2a retention in Standby mode  
 0: SRAM2a powered off in Standby mode (content is lost).  
 1: SRAM2a powered by the low-power regulator in Standby mode (content is kept).
- Bit 8 **EBORHSMPSFB**: Enable BORH and SMPS step-down converter forced in Bypass interrupts for CPU1  
 0: Interrupts BORHF and SMPSFB to CPU1 disabled  
 1: interrupts BORHF and SMPSFB to CPU1 enabled
- Bits 7:5 Reserved, must be kept at reset value.
- Bit 4 **EWUP5**: Enable Wakeup pin WKUP5 for CPU1 (available on STM32WB55xx only)  
 When this bit is set, the external wakeup pin WKUP5 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP5 bit in the [PWR control register 4 \(PWR\\_CR4\)](#). Note that this bit is reserved in STM32WB35xx devices.
- Bit 3 **EWUP4**: Enable Wakeup pin WKUP4 for CPU1  
 When this bit is set, the external wakeup pin WKUP4 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP4 bit in the [PWR control register 4 \(PWR\\_CR4\)](#). Note that this bit is reserved in STM32WB35xx devices.
- Bit 2 **EWUP3**: Enable Wakeup pin WKUP3 for CPU1 (available on STM32WB55xx only)  
 When this bit is set, the external wakeup pin WKUP3 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP3 bit in the [PWR control register 4 \(PWR\\_CR4\)](#). Note that this bit is reserved in STM32WB35xx devices.
- Bit 1 **EWUP2**: Enable Wakeup pin WKUP2 for CPU1 (available on STM32WB55xx only)  
 When this bit is set, the external wakeup pin WKUP2 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP2 bit in the [PWR control register 4 \(PWR\\_CR4\)](#). Note that this bit is reserved in STM32WB35xx devices.
- Bit 0 **EWUP1**: Enable Wakeup pin WKUP1 for CPU1  
 When this bit is set, the external wakeup pin WKUP1 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP1 bit in the [PWR control register 4 \(PWR\\_CR4\)](#).

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

Address offset: 0x00C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C2BOOT	Res.	Res.	Res.	Res.	Res.	VBRS	VBE	Res.	Res.	Res.	WP5	WP4	WP3	WP2	WP1
rw						rw	rw				rw	rw	rw	rw	rw

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

Bit 15 **C2BOOT**: Boot CPU2 after reset or wakeup from Stop or Standby modes.

0: CPU2 does not boot after reset or wakeup from Stop or Standby modes.

1: CPU2 boots after wakeup from Stop or Standby modes.

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

Bit 9 **VBRS**: V<sub>BAT</sub> battery charging resistor selection

0: Charge V<sub>BAT</sub> through a 5 kΩ resistor

1: Charge V<sub>BAT</sub> through a 1.5 kΩ resistor

Bit 8 **VBE**: V<sub>BAT</sub> battery charging enable

0: V<sub>BAT</sub> battery charging disabled

1: V<sub>BAT</sub> battery charging enabled

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

Bit 4 **WP5**: Wakeup pin WKUP5 polarity (available on STM32WB55xx only)

This bit defines the polarity used for an event detection on external wake-up pin, WKUP5

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Note that this bit is reserved in STM32WB35xx devices.

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

This bit defines the polarity used for an event detection on external wake-up pin, WKUP4

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bit 2 **WP3**: Wakeup pin WKUP3 polarity (available on STM32WB55xx only)

This bit defines the polarity used for an event detection on external wake-up pin, WKUP3

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Note that this bit is reserved in STM32WB35xx devices.

Bit 1 **WP2**: Wakeup pin WKUP2 polarity (available on STM32WB55xx only)

This bit defines the polarity used for an event detection on external wake-up pin, WKUP2

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Note that this bit is reserved in STM32WB35xx devices.

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

This bit defines the polarity used for an event detection on external wake-up pin, WKUP1  
 0: Detection on high level (rising edge)  
 1: Detection on low level (falling edge)

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

Address offset: 0x010

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUFI	C2HF	802AF	BLE AF	CRPEF	802 WUF	BLE WUF	BORHF	SMPS FBF	Res.	Res.	WUF5	WUF4	WUF3	WUF2	WUF1
r	r	r	r	r	r	r	r	r			r	r	r	r	r

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

Bit 15 **WUFI**: Internal wakeup interrupt flag

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

Bit 14 **C2HF**: CPU2 Hold interrupt flag

This bit is set when a CPU2 wakeup is detected when C2BOOT = 0. It is cleared by PWR\_SCR.CC2HF.

Bit 13 **802AF**: 802.15.4 end of activity interrupt flag

This bit is set when a 802.15.4 activity ends. It is cleared by PWR\_SCR.C802AF.

Bit 12 **BLEAF**: BLE end of activity interrupt flag

This bit is set when a BLE activity ends. It is cleared by PWR\_SCR.CBLEAF.

Bit 11 **CRPEF**: Enable critical radio phase end of activity interrupt flag

This bit is set when Radio phase activity ends. It is cleared by PWR\_SCR.CCRPEF.

Bit 10 **802WUF** 802.15.4 wakeup interrupt flag

This bit is set when a wakeup is detected on the 802.15.4 line. It is cleared by PWR\_SCR.C802WUF.

Bit 9 **BLEWUF** BLE wakeup interrupt flag

This bit is set when a wakeup is detected on the BLE line. It is cleared by PWR\_SCR.CBLEWUF.

Bit 8 **BORHF**: BORH interrupt flag

This bit is set when the  $V_{DD}$  rises above the BORH threshold. It is cleared by PWR\_SCR.CBORHF.

Bit 7 **SMPSFBF**: SMPS step-down converter forced in Bypass interrupt flag

This bit is set when the SMPS step-down converter is enabled in SMPS mode and forced in Bypass mode due to the BORH threshold. BORH configuration by BORHC bit in [PWR control register 5 \(PWR\\_CR5\)](#) must select BORH force SMPS step-down converter in Bypass mode.

This bit is cleared by PWR\_SCR.CSMPSFBF.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **WUF5**: Wakeup flag 5 (available on STM32WB55xx only)

This bit is set when a wakeup event is detected on wakeup pin, WKUP5. It is cleared by writing '1' in the CWUF5 bit of the PWR\_SCR register. Note that this bit is reserved in STM32WB35xx devices.

Bit 3 **WUF4**: Wakeup flag 4

This bit is set when a wakeup event is detected on wakeup pin, WKUP4. It is cleared by writing '1' in the CWUF4 bit of the [PWR status clear register \(PWR\\_SCR\)](#).

Bit 2 **WUF3**: Wakeup flag 3 (available on STM32WB55xx only)

This bit is set when a wakeup event is detected on wakeup pin, WKUP3. It is cleared by writing '1' in the CWUF3 bit of the [PWR status clear register \(PWR\\_SCR\)](#). Note that this bit is reserved in STM32WB35xx devices.

Bit 1 **WUF2**: Wakeup flag 2 (available on STM32WB55xx only)

This bit is set when a wakeup event is detected on wakeup pin, WKUP2. It is cleared by writing '1' in the CWUF2 bit of the [PWR status clear register \(PWR\\_SCR\)](#). Note that this bit is reserved in STM32WB35xx devices.

Bit 0 **WUF1**: Wakeup flag 1

This bit is set when a wakeup event is detected on wakeup pin, WKUP1. It is cleared by writing '1' in the CWUF1 bit of the [PWR status clear register \(PWR\\_SCR\)](#).

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

Address offset: 0x014

Reset value: 0x0000 0002. This register is partially reset when exiting Standby/Shutdown modes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PVM03	Res.	PVM01	PVDO	VOSF	REGLPF	REGLPS	Res.	Res.	Res.	Res.	Res.	Res.	SMPSF	SMPSBF
	r		r	r	r	r	r							r	r

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

Bit 14 **PVM03**: Peripheral voltage monitoring output:  $V_{DDA}$  vs. 1.62 V

0:  $V_{DDA}$  voltage is above PVM3 threshold ( $\sim 1.62$  V)

1:  $V_{DDA}$  voltage is below PVM3 threshold ( $\sim 1.62$  V)

*Note: PVM03 is cleared when PVM3 is disabled (PVME3 = 0). After enabling PVM3, the PVM3 output is valid after the PVM3 wakeup time.*

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PVMO1**: Peripheral voltage monitoring output: V<sub>DDUSB</sub> vs. 1.2 V

- 0: V<sub>DDUSB</sub> voltage is above PVM1 threshold (~ 1.2 V)
- 1: V<sub>DDUSB</sub> voltage is below PVM1 threshold (~ 1.2 V)

*Note: PVMO1 is cleared when PVM1 is disabled (PVME1 = 0). After enabling PVM1, the PVM1 output is valid after the PVM1 wakeup time.*

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

- 0: V<sub>DD</sub> or voltage level on PVD\_IN is above the selected PVD threshold
- 1: V<sub>DD</sub> or voltage level on PVD\_IN is below the selected PVD threshold

Bit 10 **VOSF**: Voltage scaling flag

A delay is required for the internal regulator to be ready after the voltage scaling has been changed. VOSF indicates that the regulator reached the voltage level defined with VOS bits of the [PWR control register 1 \(PWR\\_CR1\)](#).

- 0: The regulator is ready in the selected voltage range
- 1: The regulator output voltage is changing to the required voltage level

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

This bit is set by hardware when the MCU is in Low-power run mode. When the MCU exits from the Low-power run mode, this bit remains at 1 until the regulator is ready in main mode. A polling on this bit must be done before increasing the product frequency.

This bit is cleared by hardware when the regulator is ready.

- 0: The regulator is ready in main mode (MR)
- 1: The regulator is in low-power mode (LPR)

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

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

- 0: The low-power regulator is not ready
- 1: The low-power regulator is ready

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **SMPSF**: SMPS step-down converter SMPS mode flag

This bit indicates that the SMPS step-down converter is in SMPS mode.

- 0: The SMPS step-down converter is not ready
- 1: The SMPS step-down converter is ready

Bit 0 **SMPSBF**: SMPS step-down converter Bypass mode flag

This bit indicates that the SMPS step-down converter is in Bypass mode.

- 0: The SMPS step-down converter is not in Bypass mode (either in Open or SMPS mode)
- 1: The SMPS step-down converter is in Bypass mode

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

Address offset: 0x018

Reset value: 0x0000 0000

Access: Three additional APB cycles are needed to write this register vs. a standard APB write.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CC2HF	C802 AF	CFBLE AF	CCRP EF	C802 WUF	CBLE WUF	CBORH F	CSMPS FBF	Res.	Res.	CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
	w	w	w	w	w	w	w	w			w	w	w	w	w

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

Bit 14 **CC2HF**: Clear CPU2 Hold interrupt flag

Setting this bit clears the C2HF flag in the PWR\_SR1. This bit is always read 0.

Bit 13 **C802AF**: Clear 802.15.4 end of activity interrupt flag

Setting this bit clears the 802AF flag in the PWR\_SR1. This bit is always read 0.

Bit 12 **CBLEAF**: Clear BLE end of activity interrupt flag

Setting this bit clears the BLEAF flag in the PWR\_SR1. This bit is always read 0.

Bit 11 **CCRPEF**: Clear critical radio phase end of activity interrupt flag

Setting this bit clears the CRPEF flag in the PWR\_SR1. This bit is always read 0.

Bit 10 **C802WUF** Clear 802.15.4 wakeup interrupt flag

Setting this bit clears the 802WUF flag in the PWR\_SR1. This bit is always read 0.

Bit 9 **CBLEWUF** Clear BLE wakeup interrupt flag

Setting this bit clears the BLEWUF flag in the PWR\_SR1. This bit is always read 0.

Bit 8 **CBORHF**: Clear BORH interrupt flag

Setting this bit clears the SBORHF flag in the PWR\_SR1. This bit is always read 0.

Bit 7 **CSMPSFBF**: Clear SMPS step-down converter forced in Bypass interrupt flag

Setting this bit clears the SMPSFBF flag in the PWR\_SR1. This bit is always read 0.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **CWUF5**: Clear wakeup flag 5 (available on STM32WB55xx only)

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

Note that this bit is reserved in STM32WB35xx devices.

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

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

Bit 2 **CWUF3**: Clear wakeup flag 3 (available on STM32WB55xx only)

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

Note that this bit is reserved in STM32WB35xx devices.

Bit 1 **CWUF2**: Clear wakeup flag 2 (available on STM32WB55xx only)

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

Note that this bit is reserved in STM32WB35xx devices.

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

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

## 6.6.8 PWR control register 5 (PWR\_CR5)

Address offset: 0x01C

Reset value: 0x0000 427X, where X is factory-programmed. This register is not reset when exiting Standby modes.

Access: Three additional APB cycles are needed to write this register vs. a standard APB write.

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

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

Bit 15 **SMPSEN**: Enable SMPS step-down converter SMPS mode enabled.

This bit is reset to 0 when SMPS step-down converter switching on the fly is enabled and the  $V_{DD}$  level drops below the BORH threshold.

0: SMPS Step down converter SMPS mode disabled.

1: SMPS Step down converter SMPS mode enabled.

**Note:** *The noise performance of analog signals may be impacted by the integrated SMPS switching. To prevent this the SMPS may be switched off on the fly when measuring analog signals.*

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

Bit 8 **BORHC**: BORH configuration selection

0: BORH generates a system reset.

1: BORH forces SMPS step-down converter Bypass mode (BORL still generates a system reset).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SMPSSC**: SMPS step-down converter supply startup current selection

Startup current is limited to maximum 80 mA + SMPSSC x 20 mA.

Bit 3:0 **SMPSVOS**: SMPS step-down converter voltage output scaling

These bits are initialized after Option byte loading with factory trimmed value to reach 1.5 V, and can subsequently be overwritten by firmware.

SMPS step down output voltage step size is 50 mV.

If factory trimmed value - 0x8 gives 1.50 V on  $V_{FBMSPS}$ , to get 1.40 V 0x2 must be subtracted from this value.

– 0x0 = minimum voltage level

– 0xF = maximum voltage level

The factory trimmed value can be obtained from address 0x1FFF 7558 bits [11:8].

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

Address offset: 0x020

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

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

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

Bits 15:0 **PUy**: Port A pull-up bit y (y=0...15)

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#). The pull-up is not activated if the corresponding PDy bit is also set.

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

Address offset: 0x024

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw															

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

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

When set, this bit activates the pull-down on PA[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#).

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

Address offset: 0x028

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

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

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

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

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#). The pull-up is not activated if the corresponding PDy bit is also set. Note that bits 15:10 are reserved in STM32WB35xx devices.

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

Address offset: 0x02C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw															

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

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

When set, this bit activates the pull-down on PB[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#). Note that bits 15:10 are reserved in STM32WB35xx devices.

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

Address offset: 0x030

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

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

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

Bits 15:0 **PUy**: Port C pull-up bit y (y=0...15)

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#). The pull-up is not activated if the corresponding PDy bit is also set. Note that bits 13:0 are reserved in STM32WB35xx devices.

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

Address offset: 0x034

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw															

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

Bits 15:0 **PDy**: Port C pull-down bit y (y=0...15)

When set, this bit activates the pull-down on PC[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#). Note that bits 13:0 are reserved in STM32WB35xx devices.

#### 6.6.15 PWR Port D pull-up control register (PWR\_PUCRD) (STM32WB55xx only)

Address offset: 0x038

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

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

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

Bits 15:0 **PUy**: Port D pull-up bit y (y=0...15)

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#). The pull-up is not activated if the corresponding PDy bit is also set.

### 6.6.16 PWR Port D pull-down control register (PWR\_PDCRD) (STM32WB55xx only)

Address offset: 0x03C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw															

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

Bits 15:0 **PDy**: Port D pull-down bit y (y=0...15)

When set, this bit activates the pull-down on PD[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#).

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

Address offset: 0x040

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PU4	PU3	PU2	PU1	PU0										
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PUy**: Port E pull-up bit y (y=0...4)

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#). The pull-up is not activated if the corresponding PDy bit is also set. Note that bits 3:0 are reserved in STM32WB35xx devices.

### 6.6.18 PWR Port E pull-down control register (PWR\_PDCRE)

Address offset: 0x044

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PD4	PD3	PD2	PD1	PD0										
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PDy**: Port E pull-down bit y (y=0...4)

When set, this bit activates the pull-down on PE[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#). Note that bits 3:0 are reserved in STM32WB35xx devices.

### 6.6.19 PWR Port H pull-up control register (PWR\_PUCRH)

Address offset: 0x058

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the RCC\_APB1RSTR1 register.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	PU3	Res.	PU1	PU0												
													rw		rw	rw

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

Bit 3 **PU3**: Port H pull-up bit 3

When set, this bit activates the pull-up on PH[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#).

The pull-up is not activated if the corresponding PDy bit is also set.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **PUy**: Port H pull-up bit y (y=0...1) (available on STM32WB55xx only)

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#). The pull-up is not activated if the corresponding PDy bit is also set. Note that bits 1:0 are reserved in STM32WB35xx devices.

## 6.6.20 PWR Port H pull-down control register (PWR\_PDCRH)

Address offset: 0x05C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the RCC\_APB1RSTR1 register.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PD3	Res.	PD1	PD0											
												rw		rw	rw

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

Bit 3 **PD3**: Port H pull-down bit 3

When set, this bit activates the pull-down on PH[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#).

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **PDy**: Port H pull-down bit y (y=0...1) (available on STM32WB55xx only)

When set, this bit activates the pull-down on PH[y] when one of the APC bits is set in [PWR control register 3 \(PWR\\_CR3\)](#) and in [PWR CPU2 control register 3 \(PWR\\_C2CR3\)](#). Note that bits 1:0 are reserved in STM32WB35xx devices.

## 6.6.21 PWR CPU2 control register 1 (PWR\_C2CR1)

Address offset: 0x080

Reset value: 0x0000 0000. This register is reset after wakeup from Standby mode, except for bits [2:0].

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

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

Bit 15 **802EWKUP**: 802.15.4 external wakeup signal

When set this bit forces a wakeup of the 802.15.4 controller. It is automatically reset when 802.15.4 controller exits its sleep mode.

0: No action.

1: Wakeup 802.15.4 controller from its sleep mode.

Bit 14 **BLEEWKUP**: BLE external wakeup

When set this bit forces a wakeup of the BLE controller. It is automatically reset when BLE controller exits its sleep mode.

0: No action

1: Wakeup BLE controller from its sleep mode

Bits 13:6 Reserved, must be kept at reset value.

Bit 5 **FPDS**: Flash memory power down mode during LPSleep for CPU2

This bit selects whether the flash memory is in power down mode or idle mode when both CPUs are in Sleep mode. Flash memory is set in power down mode only when the system is in LPSleep mode and the PWR\_CR1.FPDS bit from CPU1 also allows so.

0: Flash memory in Idle mode when system is in LPSleep mode

1: Flash memory in power down mode when system is in LPSleep mode

Bit 4 **FPDR**: Flash memory power down mode during LPRun for CPU2

This bit can only be written to 1 after unlocking this register bit, by first writing (code 0xC1B0) into this register (when writing the code register bits are not updated). Selects whether the flash memory is in power down mode or idle mode when in LPRun mode. (flash memory can only be in power down mode when code is executed from SRAM). Flash memory is set in power down mode only when the system is in LPRun mode, and the PWR\_CR1.FPDR bit from CPU1 also allows so.

0: Flash memory in Idle mode when system is in LPRun mode

1: Flash memory in power down mode when system is in LPRun mode

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LPMS[2:0]**: Low-power mode selection for CPU2

These bits are not reset when exiting Standby mode.

These bits select the low-power mode entered when CPU2 enters the deepsleep mode. The system low-power mode entered depend also on the PWR\_CR1.LPMS allowed low-power mode from CPU1.

000: Stop0 mode

001: Stop1 mode

010: Stop2 mode

011: Standby mode

1xx: Shutdown mode

*Note: If LPR bit is set, Stop2 mode cannot be selected and Stop1 mode must be entered instead of Stop2.*

*In Standby mode, SRAM2 can be preserved or not, depending on RRS bit configuration in PWR control register 3 (PWR\_CR3).*

## 6.6.22 PWR CPU2 control register 3 (PWR\_C2CR3)

Address offset: 0x084

Reset value: 0x0000 8000. This register is not reset when exiting Standby modes.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EI WUL	Res.	Res.	APC	Res.	E802 WUP	EBLE WUP	Res.	Res.	Res.	Res.	EWUP5	EWUP4	EWUP3	EWUP2	EWUP1
rw			rw		rw	rw					rw	rw	rw	rw	rw

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

Bit 15 **EIWUL**: Enable internal wakeup line for CPU2

0: Internal wakeup line to CPU2 disabled

1: Internal wakeup line to CPU2 enabled

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **APC**: Apply pull-up and pull-down configuration for CPU2

When this bit for CPU2 or the PWR\_CR3.APC bit for CPU1 is set, the I/O pull-up and pull-down configurations defined in the PWR\_PUCRx and PWR\_PDCRx registers are applied.

When both bits are cleared, the PWR\_PUCRx and PWR\_PDCRx registers are not applied to the I/Os.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **E802WUP** Enable 802.15.4 host wakeup interrupt for CPU2

0: Interrupt to CPU2 disabled

1: interrupt to CPU2 enabled

Bit 9 **EBLEWUP** Enable BLE host wakeup interrupt for CPU2

0: Interrupt to CPU2 disabled

1: interrupt to CPU2 enabled

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **EWUP5**: Enable Wakeup pin WKUP5 for CPU2 (available on STM32WB55xx only)

When this bit is set, the external wakeup pin WKUP5 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU2. The active edge is configured via the WP5 bit in the *PWR control register 4 (PWR\_CR4)*. Note that this bits is reserved in STM32WB35xx devices.

Bit 3 **EWUP4**: Enable Wakeup pin WKUP4 for CPU2

When this bit is set, the external wakeup pin WKUP4 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU2. The active edge is configured via the WP4 bit in the *PWR control register 4 (PWR\_CR4)*.

Bit 2 **EWUP3**: Enable Wakeup pin WKUP3 for CPU2 (available on STM32WB55xx only)

When this bit is set, the external wakeup pin WKUP3 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU2. The active edge is configured via the WP3 bit in the *PWR control register 4 (PWR\_CR4)*. Note that this bits is reserved in STM32WB35xx devices.

Bit 1 **EWUP2**: Enable Wakeup pin WKUP2 for CPU2 (available on STM32WB55xx only)

When this bit is set, the external wakeup pin WKUP2 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU2. The active edge is configured via the WP2 bit in the *PWR control register 4 (PWR\_CR4)*. Note that this bits is reserved in STM32WB35xx devices.

Bit 0 **EWUP1**: Enable Wakeup pin WKUP1 for CPU2

When this bit is set, the external wakeup pin WKUP1 is enabled and triggers an interrupt and wakeup from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU2. The active edge is configured via the WP1 bit in the *PWR control register 4 (PWR\_CR4)*.

### 6.6.23 PWR extended status and status clear register (PWR\_EXTSCR)

Address offset: 0x088

Reset value: 0x0000 0000

Access: Three additional APB cycles are needed to write this register vs. a standard APB write.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C2DS	C1DS	CRPF	Res.	C2STOPF	C2SBF	C1STOPF	C1SBF	Res.	Res.	Res.	Res.	Res.	CCRPF	C2CSSF	C1CSSF
r	r	r		r	r	r	r						w	w	w

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

Bit 15 **C2DS**: CPU2 deepsleep mode

This bit is set by hardware when CPU2 enters deepsleep mode or is hold by C2BOOT.

0: CPU2 is running or in sleep

1: CPU2 is in deepsleep or hold by C2BOOT

Bit 14 **C1DS**: CPU1 deepsleep mode

This bit is set by hardware when CPU1 enters deepsleep mode.

0: CPU1 is running or in sleep

1: CPU1 is in deepsleep

Bit 13 **CRPF**: Critical Radio system phase

This bit is set by hardware when the Radio system wakes up. It is reset either by hardware when the Radio system enters Low-power mode or by software when writing CCRPF.

0: Not a critical Radio system phase

1: Critical Radio system phase ongoing

Bit 12 Reserved, must be kept at reset value.

Bit 11 **C2STOPF**: System Stop flag for CPU2.

This bit is set by hardware and cleared only by any reset or by setting C2CSSF bit.

0: System has not been in Stop mode

1: System has been in Stop mode

Bit 10 **C2SBF**: System Standby flag for CPU2.

This bit is set by hardware and cleared only by a POR reset or by setting C2CSSF bit.

0: System has not been in Standby mode

1: System has been in Standby mode

Bit 9 **C1STOPF**: System Stop flag for CPU1.

This bit is set by hardware and cleared only by any reset or by setting C1CSSF bit.

0: System has not been in Stop mode

1: System has been in Stop mode

Bit 8 **C1SBF**: System Standby flag for CPU1.

This bit is set by hardware and cleared only by a POR reset or by setting C1CSSF bit.

0: System has not been in Standby mode

1: System has been in Standby mode

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

Bit 2 **CCRPF**: Clear critical Radio system phase

Setting this bit clears the CRPF bit.

Bit 1 **C2CSSF**: Clear CPU2 Stop Standby flags

Setting this bit clears the C2STOPF and C2SBF bits.

Bit 0 **C1CSSF**: Clear CPU1 Stop Standby flags

Setting this bit clears the C1STOPF and C1SBF bits.

### 6.6.24 PWR register map and reset value table

Table 35. PWR register map and reset value table

Offset	Register	Reset
0x000	PWR_CR1	31
	Reset value	30
0x004	PWR_CR2	29
	Reset value	28
0x008	PWR_CR3	27
	Reset value	26
0x00C	PWR_CR4	25
	Reset value	24
0x010	PWR_SR1	23
	Reset value	22
0x014	PWR_SR2	21
	Reset value	20
0x018	PWR_SCR	19
	Reset value	18
0x01C	PWR_CR5	17
	Reset value	16
0x020	PWR_PUCRA	15
	Reset value	14
0x024	PWR_PDCRA	13
	Reset value	12
0x028	PWR_PUCRB	11
	Reset value	10
0x02C	PWR_PDCRB	9
	Reset value	8
0x030	PWR_PUCRC	7
	Reset value	6
0	PU15	0
0	PU14	0
0	PU13	0
0	PU12	0
0	PU11	0
0	PU10	0
0	PU9	0
0	PU8	0
0	PU7	0
0	PU6	0
0	PU5	0
0	PU4	0
0	PU3	0
0	PU2	0
0	PU1	0
0	PU0	0

Table 35. PWR register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x034	PWR_PDCRC	Res.																															
	Reset value	Res.																															
0x038	PWR_PUCRD	Res.																															
	Reset value	Res.																															
0x03C	PWR_PDCRD	Res.																															
	Reset value	Res.																															
0x040	PWR_PUCRE	Res.																															
	Reset value	Res.																															
0x044	PWR_PDCRE	Res.																															
	Reset value	Res.																															
0x058	PWR_PUCRH	Res.																															
	Reset value	Res.																															
0x05C	PWR_PDCRH	Res.																															
	Reset value	Res.																															
0x080	PWR_C2CR1	Res.																															
	Reset value	Res.																															
0x084	PWR_C2CR3	Res.																															
	Reset value	Res.																															
0x088	PWR_EXTSCR	Res.																															
	Reset value	Res.																															

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 7 Peripherals interconnect matrix

### 7.1 Introduction

Several peripherals have direct connections, enabling autonomous communication and/or synchronization between them. This saves CPU resources and, consequently, reduces power consumption. In addition, these hardware connections remove software latency and result in more predictable system design.

Depending on peripherals, these interconnections can operate in Run, Sleep, Low-power run and sleep, Stop 0, Stop 1 and Stop 2 modes.

### 7.2 Interconnect matrix implementation

Table 36. Interconnect matrix implementation

Feature	STM32WB55xx	STM32WB35xx
RTC_TAMP1	X	-
RTC_TAMP2	X	X
RTC_TAMP3	X	-

### 7.3 Connection summary

Table 37. Peripherals interconnect matrix<sup>(1) (2)</sup>

Source	Destination									
	TIM1	TIM2	TIM16	TIM17	LPTIM1	LPTIM2	ADC1	COMP1	COMP2	IRTIM
TIM1	-	1	-	-	-	-	2	6	6	-
TIM2	1	-	-	-	-	-	2	6	6	-
TIM16	-	-	-	-	-	-	-	-	-	11
TIM17	1	-	-	-	-	-	-	-	-	11
LPTIM1	-	-	-	-	-	-	-	-	-	-
LPTIM2	-	-	-	-	-	-	-	-	-	-
ADC1	3	-	-	-	-	-	-	-	-	-
T. Sensor	-	-	-	-	-	-	8	-	-	-
VBAT	-	-	-	-	-	-	8	-	-	-
VREFINT	-	-	-	-	-	-	8	-	-	-
HSE	-	-	-	4	-	-	-	-	-	-
LSE	-	4	4	-	-	-	-	-	-	-
MSI	-	-	-	4	-	-	-	-	-	-
LSI	-	-	4	-	-	-	-	-	-	-

Table 37. Peripherals interconnect matrix<sup>(1)</sup> <sup>(2)</sup> (continued)

Source	Destination									
	TIM1	TIM2	TIM16	TIM17	LPTIM1	LPTIM2	ADC1	COMP1	COMP2	IRTIM
<b>MCO</b>	-	-	-	<b>4</b>	-	-	-	-	-	-
<b>EXTI</b>	-	-	-	-	-	-	<b>2</b>	-	-	-
<b>RTC</b>	-	-	<b>4</b>	-	<b>5</b>	<b>5</b>	-	-	-	-
<b>COMP1</b>	<b>9</b>	<b>9</b>	<b>9</b>	<b>9</b>	<b>5</b>	<b>5</b>	-	-	-	-
<b>COMP2</b>	<b>9</b>	<b>9</b>	<b>9</b>	<b>9</b>	<b>5</b>	<b>5</b>	-	-	-	-
<b>SYST ERR</b>	<b>10</b>	-	<b>10</b>	<b>10</b>	-	-	-	-	-	-
<b>USB</b>	-	<b>7</b>	-	-	-	-	-	-	-	-

1. Numbers in table are links to corresponding subsections of [Section 7.4: Interconnection details](#).
2. The “-” symbol in grayed cells means no interconnect.

## 7.4 Interconnection details

### 7.4.1 From timer (TIM1/TIM2/TIM17) to timer (TIM1/TIM2)

#### Purpose

Some of the timers are linked together internally for synchronization or chaining.

When one timer is configured in Master Mode, it can reset, start, stop or clock the counter of another timer configured in Slave Mode. A description of the feature is provided in [Section 24.3.26: Timer synchronization](#).

The modes of synchronization are detailed in:

- [Section 24.3.26: Timer synchronization](#) for advanced-control timers (TIM1)
- [Section 25.3.18: Timers and external trigger synchronization](#) for general-purpose timers (TIM2)

#### Triggering signals

The output (from Master) is on signal TIMx\_TRGO (and TIMx\_TRGO2 for TIM1) following a configurable timer event. The input (to slave) is on signals TIMx\_ITR0/ITR1/ITR2/ITR3.

The input and output signals for TIM1 are shown in [Figure 144: Advanced-control timer block diagram](#).

The possible master/slave connections are given in:

- [Table 162: TIM1 internal trigger connection](#)
- [Table 166: TIM2 internal trigger connection](#)

#### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.

## 7.4.2 From timer (TIM1/TIM2) and EXTI to ADC (ADC1)

### Purpose

General-purpose timer TIM2, advanced-control timer TIM1 and EXTI can be used to generate an ADC triggering event.

TIMx synchronization is described in [Section 24.3.27: ADC synchronization](#).

ADC synchronization is described in [Section 15.4: Conversion on external trigger and trigger polarity \(EXTSEL, EXTEN\)](#).

### Triggering signals

The output (from timer) is on signal TIMx\_TRGO, TIMx\_TRGO2 or TIMx\_CCx event.

The input (to ADC) is on signals EXT[15:0] and JEXT[15:0].

The connection between timers and ADC is provided in:

- [Table 79: ADC1 - External triggers for regular channels](#)
- [Table 80: ADC1 - External trigger for injected channels](#)

### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.

## 7.4.3 From ADC (ADC1) to timer (TIM1)

### Purpose

ADC1 can provide trigger event through watchdog signals to advanced-control timers (TIM1).

A description of the ADC analog watchdog setting is provided in [Section 15.7: Analog window watchdog](#).

Trigger settings on the timer are provided in [Section 24.3.4: External trigger input](#).

### Triggering signals

The output (from ADC) is on signals ADC\_AWDx\_OUT n = 1, 2, 3 (for ADC1) x = 1, 2, 3 (three watchdogs on ADC) and the input (to timer) on signal TIMx\_ETR (external trigger).

### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.

## 7.4.4 From HSE, LSE, LSI, MSI, MCO, RTC to timers (TIM2/TIM16/TIM17)

### Purpose

External clocks (HSE, LSE), internal clocks (LSI, MSI), microcontroller output clock (MCO), GPIO and RTC wakeup interrupt can be used as input to general-purpose timers (TIM16/17) channel 1.

This makes possible calibration of the HSI16/MSI system clocks (with TIM16 and LSE) or of the LSI (with TIM16 and HSE). This is also used to precisely measure LSI (with TIM16 and HSI16) or MSI (with TIM17 and HSI16) oscillator frequency.

When Low Speed External (LSE) oscillator is used, no additional hardware connections are required.

This feature is described in [Section 8.2.21: Internal/external clock measurement with TIM16/TIM17](#).

External clock LSE can be used as input to general-purpose timer (TIM2) on TIM2\_ETR pin, see [Section 8.2.21: Internal/external clock measurement with TIM16/TIM17](#)..

### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.

## 7.4.5 From RTC, COMP1, COMP2 to low-power timers (LPTIM1/LPTIM2)

### Purpose

RTC alarm A/B, RTC\_TAMP1/2/3 input detection, COMP1/2\_OUT can be used as trigger to start LPTIM counters (LPTIM1/2).

### Triggering signals

This trigger feature is described in [Section 27.4.6: Trigger multiplexer](#) (and following sections).

The input selection is described in [Table 174: LPTIM1 external trigger connection](#) and [Table 175: LPTIM2 external trigger connection](#).

### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep, Stop 0, Stop 1, Stop 2 (LPTIM1 only).

## 7.4.6 From timer (TIM1/TIM2) to comparators (COMP1/COMP2)

### Purpose

Advanced-control timer (TIM1) and general-purpose timer (TIM2) can be used as blanking window input to COMP1/COMP2.

The blanking function is described in [Section 18.3.7: Comparator output blanking function](#).

The blanking sources are given in

- [Section 18.6.1: Comparator 1 control and status register \(COMP1\\_CSR\)](#) bits 20:18  
BLANKING[2:0]
- [Section 18.6.2: Comparator 2 control and status register \(COMP2\\_CSR\)](#) bits 20:18  
BLANKING[2:0]

### Triggering signals

Timer output signal TIMx\_Ocx are the inputs to blanking source of COMP1/COMP2.

### Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.

### 7.4.7 From USB to timer (TIM2)

#### Purpose

USB (FS SOF) can generate a trigger to general-purpose timer (TIM2).

Connection of USB to TIM2 is described in [Table 166: TIM2 internal trigger connection](#).

#### Triggering signals

Internal signal generated by USB\_FS Start Of Frame.

#### Active power mode(s)

Run, Sleep.

### 7.4.8 From internal analog to ADC1

#### Purpose

Internal temperature sensor (VTS), internal reference voltage (VREFINT) and VBAT monitoring channel are connected to ADC1 input channel.

This is according to:

- [Section 15.2: ADC main features](#)
- [Section 15.3.8: Channel selection \(CHSEL, SCANDIR, CHSELRMOD\)](#)
- [Section 15.8: Temperature sensor and internal reference voltage](#)
- [Section 15.9: Battery voltage monitoring](#)

#### Active power mode(s)

Run, Sleep, low-power run, Low-power sleep.

### 7.4.9 From comparators (COMP1/COMP2) to timers (TIM1/TIM2/TIM16/TIM17)

#### Purpose

Comparators (COMP1/COMP2) output values can be connected to timers TIM1/TIM2/TIM16/TIM17 input captures or TIMx\_ETR signals.

The connection to ETR is described in [Section 24.3.4: External trigger input](#).

Comparators (COMP1/COMP2) output values can also generate break input signals for timer TIM1 on input pins TIMx\_BKIN or TIMx\_BKIN2 through GPIO alternate function selection using open drain connection of I/Os.

The possible connections are given in:

- [Section 24.4.23: TIM1 option register 1 \(TIM1\\_OR1\)](#)
- [Section 24.4.28: TIM1 Alternate function register 2 \(TIM1\\_AF2\)](#)
- [Section 25.4.22: TIM2 option register 1 \(TIM2\\_OR1\)](#)
- [Section 24.4.27: TIM1 alternate function option register 1 \(TIM1\\_AF1\)](#)
- [Section 26.2: TIM16/TIM17 main features](#)

**Active power mode(s)**

Run, Sleep, Low-power run, Low-power sleep.

**7.4.10 From system errors to timers (TIM1/TIM16/TIM17)****Purpose**

CSS, CPU hard fault, RAM parity error, FLASH ECC double error detection, PVD can generate system errors in the form of timer break toward timers TIM1/TIM16/TIM17.

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

List(s) of possible break source(s) are described in:

- [\*Section 24.3.16: Using the break function\*](#) (TIM1)
- [\*Section 26.3.11: Using the break function\*](#) (TIM16/TIM17)
- [\*Figure 253: TIM16/TIM17 block diagram\*](#)

**Active power mode(s)**

Run, Sleep, Low-power run, Low-power sleep.

**7.4.11 From timers (TIM16/TIM17) to IRTIM****Purpose**

General-purpose timer (TIM16/TIM17) output channel TIMx\_OC1 are used to generate the waveform of infrared signal output.

The functionality is described in [\*Section 28: Infrared interface \(IRTIM\)\*](#).

**Active power mode(s)**

Run, Sleep, Low-power run, Low-power sleep.

## 8 Reset and clock control (RCC)

### 8.1 Reset

There are three types of reset, namely:

1. a system reset
2. a power reset
3. a backup domain reset

#### 8.1.1 Power reset

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

1. a brown-out reset (BOR)
2. when exiting from Standby mode
3. when exiting from Shutdown mode

A brown-out reset, including power-on or power-down reset (POR/PDR), sets all registers to their reset values except the Backup domain.

When exiting Standby mode, all registers in the  $V_{CORE}$  domain are set to their reset value. Registers outside the  $V_{CORE}$  domain (RTC, WKUP, IWDG, and Standby/Shutdown modes control) are not impacted.

When exiting Shutdown mode, a Brown-out reset is generated, resetting all registers except those in the Backup domain.

#### 8.1.2 System reset

A system reset sets all registers to their reset values except register (RCC\_CSR) and the registers in the Backup domain.

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

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

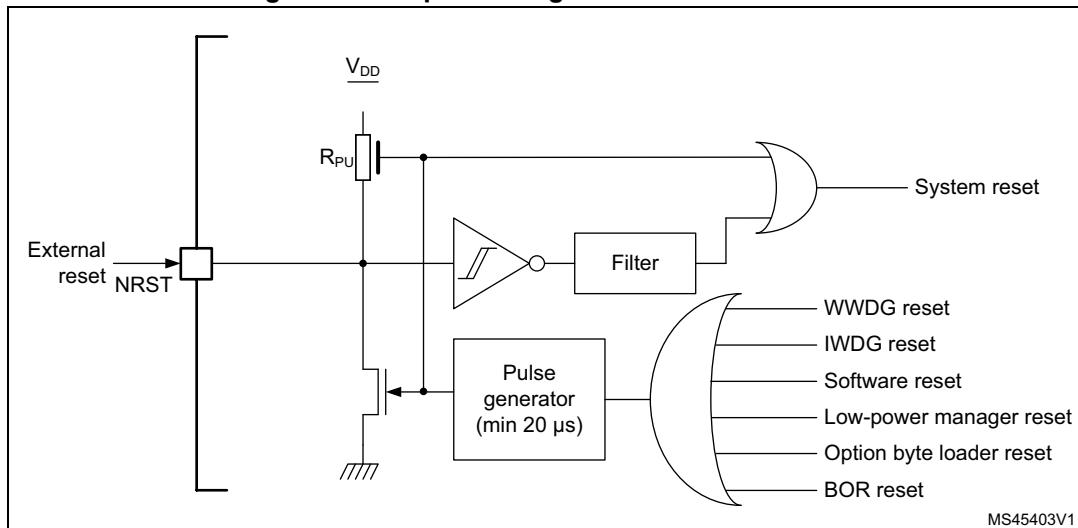
The reset source can be identified by checking the reset flags in the [RCC control/status register \(RCC\\_CSR\)](#).

These sources act on the NRST pin, always kept low during the delay phase. The CPU1 RESET service routine vector is selected via the BOOT0 and BOOT1.

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

In case of an internal reset, the internal pull-up  $R_{PU}$  is deactivated to reduce power consumption through the pull-up resistor.

**Figure 14. Simplified diagram of the reset circuit**



## Software reset

The SYSRESETREQ bit in CPU1 application interrupt and reset control register may be set to force a software reset on the device (refer to PM0214 “STM32 Cortex®-M4 MCUs and MPUs programming manual”, available on [www.st.com](http://www.st.com)).

The SYSRESETREQ bit in CPU2 application interrupt and reset control register may be set to force a software reset on the device.

## Low-power mode security reset

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

1. When entering Standby mode: reset is enabled by resetting nRST\_STDBY bit in User option Bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
  2. When entering Stop mode: reset is enabled by resetting nRST\_STOP bit in User option bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.
  3. When entering Shutdown mode: reset is enabled by resetting nRST\_SHDW bit in User option bytes. In this case, whenever a Shutdown mode entry sequence is successfully executed, the device is reset instead of entering Shutdown mode.

For further information on the User option bytes refer to [Section 3.4.1: Option bytes description](#).

## Option byte loader reset

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

### 8.1.3 Backup domain reset

The backup domain has two specific resets.

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.

A backup domain reset only affects the LSE oscillator, the RTC, the Backup registers and the RCC Backup domain control register.

## 8.2 Clocks

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

- HSI16 (high speed internal) 16 MHz RC oscillator clock
- MSI (multispeed internal) RC oscillator clock from 100 kHz to 48 MHz.
- HSE 32 MHz oscillator clock
- PLL clock

The MSI is used as system clock source after startup from Reset, configured at 4 MHz.

The devices have the following additional clock sources:

- LSI1: 32 kHz low speed internal RC, which may drive the independent watchdog and optionally the RTC used for Auto-wakeup from Stop and Standby modes (shall not be used for RF system Auto-wakeup).
- LSI2: 32 kHz low speed low drift internal RC, which may drive the independent watchdog and optionally the RTC used for Auto-wakeup from Stop and Standby modes.
- LSE: 32.768 kHz low speed external crystal, which optionally drives the RTC used for Auto-wakeup or the RF system Auto-wakeup from Stop and Standby modes, or the real-time clock (RTCCLK).
- HSI48: RC 48 MHz internal clock sources to potentially drive the USB Full Speed and the true RNG.

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

Several prescalers can be used to configure the AHB frequencies (HCLK1, HCLK2 and HCLK4) the high speed APB (PCLK2) and the low speed APB (PCLK1) domains. The maximum frequency of the AHB (HCLK1 and HCLK4), and of the PCLK1 and PCLK2 domains is 64 MHz. The maximum frequency of the AHB (HCLK2) domain is 32 MHz.

Most peripheral clocks are derived from their bus clock (HCLK, PCLK) except:

- The 48 MHz clock, used for USB FS and true RNG. This clock is derived (selected by software) from one of the following sources:

- PLL VCO (PLLQCLK only available in Run mode)
- PLLSAI1 VCO (PLLSAI1QCLK only available in Run mode)
- MSI clock (only available in Run mode)
- HSI48 internal oscillator (only available in Run mode)

When the MSI clock is auto-trimmed with the LSE, it can be used by the USB FS device.

The HSI48 clock can be coupled to the clock recovery system allowing adequate clock connection for the USB FS (crystal less solution).

- The ADCs clock, which is derived (selected by software) from one of the three following sources:

- system clock (SYSCLK only available in Run mode)
- PLL VCO (PLLAPCLK only available in Run mode)
- PLLSAI1 VCO (PLLSAI1RCLK only available in Run mode)

- The U(S)ARTs clocks, which are derived (selected by software) from one of the four following sources:

- system clock (SYSCLK only available in Run mode)
- HSI16 clock (only available in Run and Stop modes)
- LSE clock (only available in Run and Stop modes)
- APB clock (PCLK depending on which APB is mapped the U(S)ART only available in CRun when enabled in U(S)ARTxEN and CSleep when also enabled in U(S)ARTxSMEN)

The wakeup from Stop mode is supported only when the clock is HSI16 or LSE.

- The I<sup>2</sup>Cs clocks, derived (selected by software) from one of the three following sources:

- system clock (SYSCLK only available in Run mode)
- HSI16 clock (only available in Run and Stop modes)
- APB clock (PCLK depending on which APB is mapped the I<sup>2</sup>C only available in CRun when enabled in I2CxEN and CSleep when also enabled in I2CxSMEN)

The wakeup from Stop mode is supported only when the clock is HSI.

- The SAI1 clock, which is derived (selected by software) from one of the following sources:

- an external clock mapped on SAI1\_EXTCLK for SAI1.
- PLL VCO (PLLPCLK only available in Run mode)
- PLLSAI1 VCO (PLLSAI1PCLK only available in Run mode)
- HSI16 clock (only available in Run mode)

The SAI1 can use the HSI16 clock when it is master just to detect an audio activity on the data line, in reception mode. Potentially, it allows to reduce power consumption without having to switch ON the PLL when there is no audio data flow in reception.

- The low-power timer (LPTIMx) clocks, which are derived (selected by software) from one of the five following sources:

- LSI clock (LSI1 or LSI2 only available in Run and Stop modes)

- LSE clock (only available in Run and Stop modes)
- HSI16 clock (only available in Run mode)
- APB clock (PCLK depending on which APB is mapped the LPTIMx only available in CRun when enabled in LPTIMxEN and CSleep when also enabled in LPTIMxSMEN)
- External clock mapped on LPTIMx\_IN1 (only available in Run and Stop modes)  
The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE, or in external clock mode.
- The RTC and LCD clock, which is derived (selected by software) from one of the three following sources:
  - LSE clock
  - LSI clock (LSI1 or LSI2)
  - HSE clock divided by 32The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE.
- The IWDG clock, which is always the LSI clock (LSI1 or LSI2).
- The RF system wakeup clock, which is derived (selected by software) from one of the two following sources:
  - LSE clock
  - HSE clock divided by 1024The functionality in Stop mode (including wakeup) is supported only when the clock is LSE. When HSE/1024 is selected as RF system wakeup source HSEON bit must be set by the application.
- The RF system clock is derived (selected by hardware) from one of the two following sources:
  - HSI16 clock
  - HSE clockThe functionality in Stop mode is supported only when the clock is HSI16 is selected as wakeup clock by STOPWUCK.

The RCC feeds the CPU1 system timer (SysTick) external clock with the AHB clock (HCLK1) divided by 8. The SysTick can work either with this clock or directly with the CPU1 clock (HCLK1), configurable in the SysTick Control and status register.

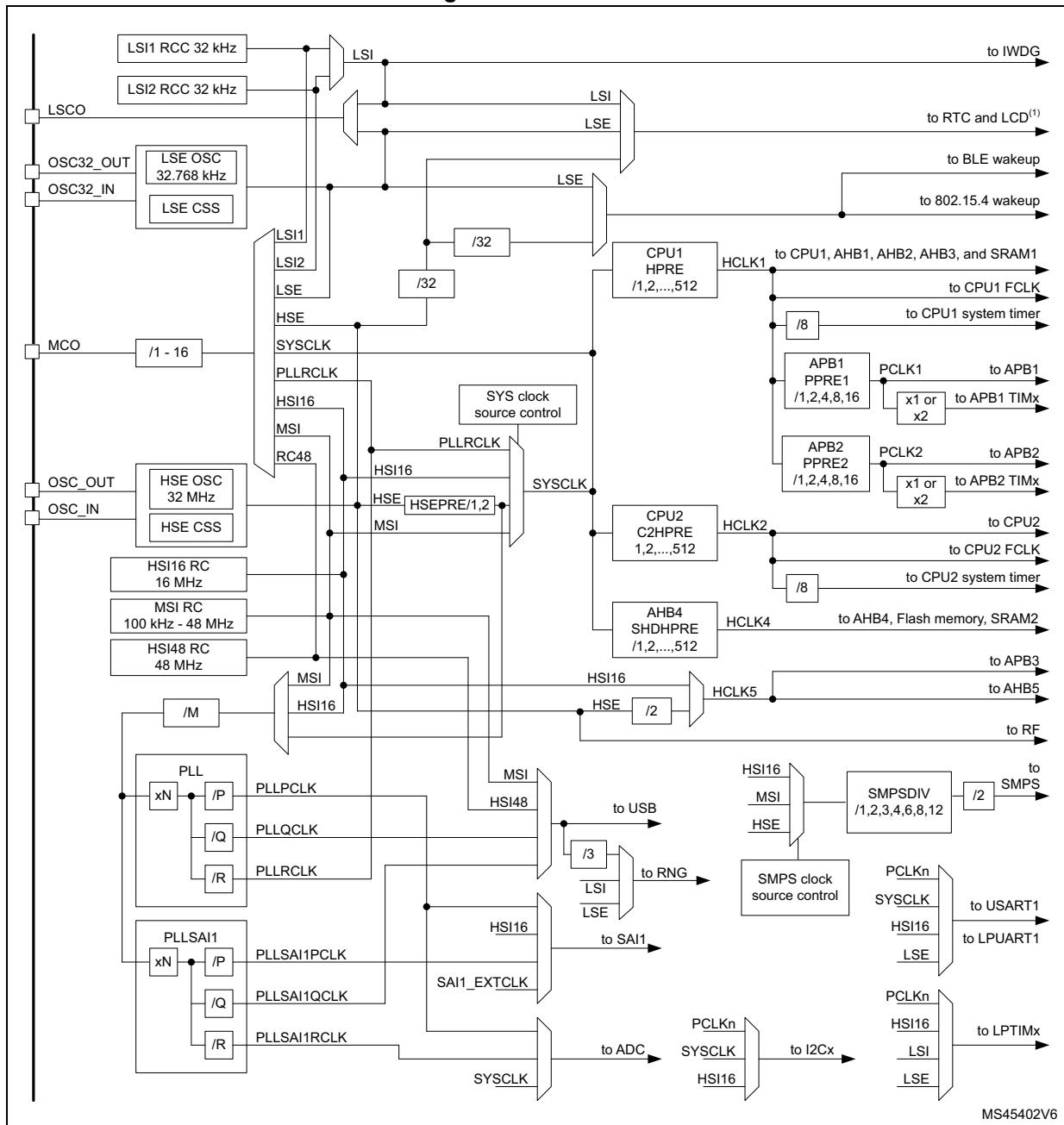
FCLK1 acts as CPU1 free-running clock. For more details refer to PM0214.

The RCC feeds the CPU2 system timer (SysTick) external clock with the AHB clock (HCLK2) divided by 8. The SysTick can work either with this clock or directly with the CPU2 clock (HCLK2), configurable in the SysTick Control and status Register.

FCLK2 acts as CPU2 free-running clock.

The clock tree is detailed in [Figure 15](#), where blue is used for system clock, synchronous clock for CPUs and shared buses, and red is used for asynchronous Radio peripheral bus clock.

Figure 15. Clock tree



1. LCD is available only on STM32WB55xx devices.
2. For full details about the internal and external clock source characteristics refer to the “Electrical characteristics” section in the device datasheet.
3. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). When the programmable factor is 1, the AHB prescaler must be equal to 1.

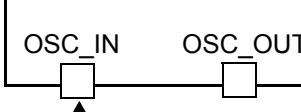
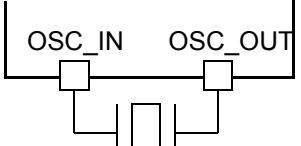
## 8.2.1 HSE clock

The high speed external clock signal (HSE) can be generated from two possible clock sources (see [Figure 16](#)):

- HSE external crystal
- HSE user external clock

The crystal has to be placed as close as possible to the oscillator pins to minimize output distortion and startup stabilization time. The loading capacitance are integrated and can be adjusted according to [Frequency tuning](#).

**Figure 16. HSE clock sources**

Clock source	Hardware configuration
External clock	 <p>External source</p>
Crystal	

### External crystal (HSE crystal)

The 32 MHz external oscillator has the advantage of producing a very accurate rate on the main clock, and is mandatory for any Radio operation.

The associated hardware configuration is shown in [Figure 16](#). Refer to the electrical characteristics section of the *datasheet* for more details.

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

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

### External source

In this mode, selectable by setting the HSEON bit in the [RCC clock control register \(RCC\\_CR\)](#), an external clock source must be provided, with a frequency of 32 MHz. The external clock signal (sinus) with ~45 to 55 % duty cycle (refer to the *datasheet*) has to drive the OSC\_IN pin, while the OSC\_OUT pin must be left not connected (see [Figure 16](#)).

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

## Frequency tuning

The HSE oscillator frequency can vary from one chip to another due to manufacturing process variations and used crystal. User can tune the HSE frequency in the application by writing the HSETUNE bits in [RCC clock HSE register \(RCC\\_HSECR\)](#). The HSE frequency can be measured by outputting the HSE clock on the MCO.

HSE oscillator gain and sense can be controlled by HSEGMC and HSES bits in [RCC clock HSE register \(RCC\\_HSECR\)](#). Refer to AN5042 for the HSE trimming procedure.

## 8.2.2 HSI16 clock

The HSI16 clock signal is generated from an internal 16 MHz oscillator.

The HSI16 oscillator has the advantage of providing a clock source at low cost. 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.

The HSI16 clock can be selected as system clock after wakeup from Stop modes (Stop0, Stop1 or Stop2), see [Section 8.3: Low-power modes](#). It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails, see [Section 8.2.11](#).

When the RF system is enabled the HSI16 must be selected as system clock after wakeup from Stop modes.

The HSI16 clock is used as system clock after restart wakeup from Standby.

### Calibration

RC oscillator frequencies can vary from one chip to another because of 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 internal clock sources calibration register \(RCC\\_ICSCR\)](#).

The RC oscillator speed can be affected by voltage or temperature variations, the user can trim the HSI16 frequency in the application using the HSITRIM bits in the [RCC internal clock sources calibration register \(RCC\\_ICSCR\)](#).

For more details on how to measure the HSI16 frequency variation, refer to [Section 8.2.21](#).

The HSIRDY flag in the [RCC clock control register \(RCC\\_CR\)](#) indicates if the HSI16 RC is stable or not. At startup, the HSI16 RC output clock is not released until this bit is set by hardware.

The HSI16 RC can be switched on and off using the HSION bit in the [RCC clock control register \(RCC\\_CR\)](#).

The HSI16 signal can also be used as a backup source (auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 8.2.11](#).

## 8.2.3 MSI clock

The MSI clock signal is generated from an internal RC oscillator. Its frequency range can be adjusted by software by using the MSIRANGE[3:0] bits in the [RCC clock control register \(RCC\\_CR\)](#). Twelve frequency ranges are available: 100 kHz, 200 kHz, 400 kHz, 800 kHz, 1 MHz, 2 MHz, 4 MHz (default value), 8 MHz, 16 MHz, 24 MHz, 32 MHz and 48 MHz.

The MSI clock is used as system clock after restart from Reset, wakeup from Shutdown low-power modes. After restart from Reset, the MSI frequency is set to its default value 4 MHz. Refer to [Section 8.3: Low-power modes](#).

The MSI clock can be selected as system clock after a wakeup from Stop mode (Stop 0, Stop 1 or Stop 2). Refer to [Section 8.3: Low-power modes](#). It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 8.2.11](#).

In addition, when used in PLL-mode with the LSE, it provides a very accurate clock source which can be used by the USB FS device, and feed the PLL to run the system at the maximum speed 64 MHz.

The MSIRDY flag in the [RCC clock control register \(RCC\\_CR\)](#) indicates if the MSI RC is stable or not. At startup, the MSI RC output clock is not released until this bit is set by hardware. The MSI RC can be switched on and off by using the MSION bit in the [RCC clock control register \(RCC\\_CR\)](#).

### Hardware auto calibration with LSE (PLL-mode)

When a 32.768 kHz external oscillator is present in the application, it is possible to configure the MSI in a PLL-mode by setting the MSIPLEN bit in the [RCC clock control register \(RCC\\_CR\)](#). When configured in PLL-mode, the MSI automatically calibrates itself thanks to the LSE. This mode is available for all MSI frequency ranges. At 48 MHz, the MSI in PLL-mode can be used for the USB FS device.

### Software calibration

The MSI RC oscillator frequency 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 25 °C ambient temperature ( $T_A$ ). After reset, the factory calibration value is loaded in the MSICAL[7:0] bits in the [RCC internal clock sources calibration register \(RCC\\_ICSCR\)](#). If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. The MSI frequency in the application can be trimmed by using the MSITRIM[7:0] bits in the RCC\_ICSCR register. For more details on how to measure the MSI frequency variation refer to [Section 8.2.21](#).

## 8.2.4 HSI48 clock

The HSI48 clock signal is generated from an internal 48 MHz RC oscillator and can be used directly for the USB and for the random number generator (true RNG).

The internal 48 MHz RC oscillator is mainly dedicated to provide a high precision clock to the USB peripheral by means of a special clock recovery system (CRS) circuitry. The CRS can use the USB SOF signal, the LSE or an external signal to automatically and quickly adjust the oscillator frequency on-fly. It is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, the HSI48 RC oscillator runs on its default frequency which is subject to manufacturing process variations.

For more details on how to configure and use the CRS peripheral refer to [Section 40: Clock recovery system \(CRS\)](#).

The HSI48RDY flag in the Clock recovery register (RCC\_CRRCR) indicates whether the HSI48 RC oscillator is stable or not. At startup, the HSI48 RC oscillator output clock is not released until this bit is set by hardware.

The HSI48 can be switched on and off using the HSI48ON bit in the Clock recovery register (RCC\_CRRCR).

## 8.2.5 PLLs

The device embeds the following PLLs: PLL, PLLSAI1. Each PLL provides up to three independent outputs. The internal PLLs can be used to multiply the HSI, HSE or MSI output clock frequency. The PLLs input frequency must be between 2.66 and 16 MHz. The selected clock source is divided by a programmable factor PLLM from 1 to 8 to provide a clock frequency in the requested input range. Refer to [Figure 15: Clock tree and RCC PLL configuration register \(RCC\\_PLLCFGR\)](#).

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

To modify the PLL configuration, proceed as follows:

1. Disable the PLL by setting PLLxxxON to 0 in [RCC clock control register \(RCC\\_CR\)](#).
2. Wait until PLLxxxRDY is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLxxxON to 1.
5. Enable the desired PLL outputs by configuring PLLPEN, PLLQEN, PLLREN in [RCC PLL configuration register \(RCC\\_PLLCFGR\)](#), or [RCC PLLSAI1 configuration register \(RCC\\_PLLSAI1CFGR\)](#).

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

The PLL output frequency must not exceed 64 MHz.

The enable bit of each PLL output clock (PLLPEN, PLLQEN, PLLREN) can be modified at any time without stopping the corresponding PLL. PLLREN cannot be cleared if PLLRCLK is used as system clock.

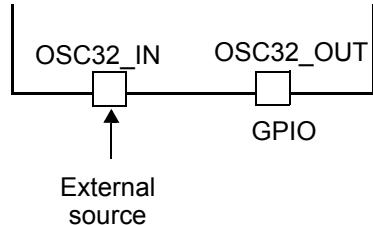
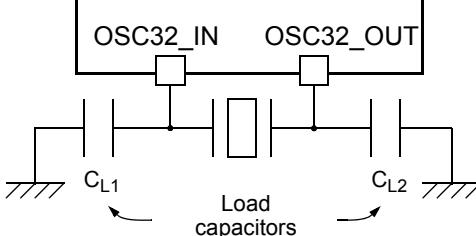
In Range 2 the PLL M divider input clock shall not exceed 16 MHz. HSEPRE bit in [RCC clock control register \(RCC\\_CR\)](#) shall select HSE divided by 2.

## 8.2.6 LSE clock

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

The 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 17. LSE clock sources

Clock source	Hardware configuration
External clock	
Crystal / ceramic resonators	

The LSE crystal is switched on and off using the LSEON bit in [RCC backup domain control register \(RCC\\_BDCR\)](#). The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the [RCC backup domain control register \(RCC\\_BDCR\)](#) to obtain the best compromise between robustness and short start-up time on one side and low-power-consumption on the other side. The LSE drive can be decreased to the lower drive capability (LSEDRV = 00) when the LSE is ON. However, once LSEDRV is selected, the drive capability cannot be increased if LSEON = 1.

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

### External source (LSE bypass)

In this mode, selectable by setting the LSEBYP and LSEON bits in the [RCC AHB1 peripheral clocks enable in Sleep modes register \(RCC\\_AHB1SMENR\)](#) an external clock source must be provided, with a frequency of up to 1 MHz. The external clock signal (square, sinus or triangle) with ~50 % duty cycle has to drive the OSC32\_IN pin while the OSC32\_OUT pin can be used as GPIO, see [Figure 17](#).

## 8.2.7 LSI1 clock

The LSI1 RC acts as a low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG), RTC, LCD and RF wakeup. The clock frequency is 32 kHz. For more details refer to the electrical characteristics section of the datasheet.

The LSI1 RC can be switched on and off using the LSI1ON bit in the [RCC control/status register \(RCC\\_CSR\)](#).

The LSI1RDY flag in the [RCC control/status register \(RCC\\_CSR\)](#) indicates if the LSI1 oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the [RCC clock interrupt enable register \(RCC\\_CIER\)](#).

### 8.2.8 LSI2 clock

The LSI2 RC acts as a low drift low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG), RTC and LCD wakeup. The clock frequency is  $\sim 32$  kHz. For more details refer to the electrical characteristics section of the datasheets.

The LSI2 RC can be switched on and off using the LSI2ON bit in the [RCC control/status register \(RCC\\_CSR\)](#).

The LSI2RDY flag in the [RCC control/status register \(RCC\\_CSR\)](#) indicates if the LSI2 oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the [RCC clock interrupt enable register \(RCC\\_CIER\)](#).

After any system reset NRST and before using the LSI2, its oscillator must get the trimming information. Firmware must copy the LSI2 trimming information from the ST production trimmed Flash memory location to the LSI2TRIM location in register RCC\_SCR.

### 8.2.9 System clock (SYSCLK) selection

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

- MSI oscillator
- HSI16 oscillator
- HSE oscillator
- PLLRCLK

The system clock maximum frequency in Range 1 is 64 MHz. After a system reset, the MSI oscillator, at 4 MHz, is selected as system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the switch will occur when the clock source becomes ready. Status bits in the [RCC internal clock sources calibration register \(RCC\\_ICSCR\)](#) indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

When waking up from Standby mode the HSI16 is selected as system clock.

In Range 2 the system clock must not exceed 16 MHz. HSEPRE bit in [RCC clock control register \(RCC\\_CR\)](#) shall select HSE divided by 2.

### 8.2.10 Clock source frequency versus voltage scaling

[Table 38](#) gives the different clock source frequencies depending on the product voltage range.

**Table 38. Maximum clock source frequency**

Product voltage range	Clock frequency			
	MSI	HSI	HSE	PLL/PLLSAI1
Range 1	48 MHz	16 MHz	32 MHz	64 MHz (VCO max = 344 MHz)
Range 2	16 MHz range	16 MHz	32 MHz <sup>(1)</sup>	16 MHz (VCO max = 128 MHz)

1. HSEPRE shall select divide by 2.

### 8.2.11 Clock security system (CSS) on HSE

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, the HSE oscillator is automatically disabled, a clock failure event is sent to the break input of the advanced-control timers (TIM1 and TIM16/17) and an interrupt is generated to inform the software about the failure (clock security system interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the CPU1 and CPU2 NMI (non-maskable interrupt) exception vector.

**Note:** *Once the HSE CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and a NMI is automatically generated. The NMI is executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the [RCC clock interrupt clear register \(RCC\\_CICR\)](#).*

If the HSE oscillator is used directly or indirectly as the system clock (indirectly means: it is used as PLL input clock, and the PLL clock is used as system clock), a detected failure causes a switch of the system clock to the MSI or the HSI16 oscillator depending on the STOPWUCK configuration in the [RCC clock configuration register \(RCC\\_CFGR\)](#), and the disabling of the HSE oscillator. If the HSE clock (divided or not) is the clock entry of the PLL used as system clock when the failure occurs, the PLL is disabled too.

### 8.2.12 Clock security system on LSE (LSECSS)

The clock security system on LSE can be activated by software writing the LSECSSON bit in the [RCC control/status register \(RCC\\_CSR\)](#). This bit can be disabled only by a hardware reset or RTC software reset, or after a failure detection on LSE. LSECSSON must be written after LSE and LSI are enabled (LSEON and LSI1ON enabled) and ready (LSERDY and LSIRDY set by hardware), and after the RTC clock has been selected by RTCSEL.

The CSS on LSE is working in all modes except VBAT. It is working also under system reset (excluding power on reset). If a failure is detected on the external 32 kHz oscillator, the LSE clock is no longer supplied to the RTC but no hardware action is made to the registers. If the MSI was in PLL-mode, this mode is disabled.

In Standby mode a wakeup is generated. In other modes an interrupt can be sent to wakeup the software (see [RCC clock interrupt enable register \(RCC\\_CIER\)](#), [RCC clock interrupt flag register \(RCC\\_CIFR\)](#), [RCC clock interrupt clear register \(RCC\\_CICR\)](#)).

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

### 8.2.13 LSI source selection

The LSI used in the system can be selected to come either from LSI1 or LSI2. Whenever LSI2 is turned on by LSI2ON register bit, the LSI2 (when ready) is selected as LSI source.

To switch from LSI2 to LSI1 without interrupting the LSI clock, the LSI1 must first be switched on by LSI1ON register bit. The FW must verify that the LSI1 is ready by LSI1RDY register bit before disabling LSI2 in LSI2ON register bit.

### 8.2.14 SMPS step-down converter clock

The SMPS step-down converter clock source can be either the MSI, HSI16 or HSE clock. It is selected by programming the SMPSEL in the *RCC SMPS step-down converter control register (RCC\_SMPSCR)*. The system must always be configured so as to get a SMPS step-down converter clock frequency between 4 and 8 MHz. The appropriate division factor shall be set in SMPSDIV. There is a further fixed division by 2 before driving the clock to the SMPS step-down converter.

After a system reset, the HSI16 is selected as SMPS step-down converter clock.

When entering low power modes Stop1, Stop2, Standby, or Shutdown the SMPS step-down converter clock source selected in SMPSEL in the *RCC SMPS step-down converter control register (RCC\_SMPSCR)* must be set to select the same source as the system clock in SW in the *RCC clock configuration register (RCC\_CFGR)*.

When waking up from Standby mode and powering on the  $V_{CORE}$  supply, the HSI16 is selected as SMPS step-down converter clock, independent from the selection in SMPSEL.

When the Radio system is active the HSE is selected as SMPS step-down converter clock, independent from the selection in SMPSEL.

The SMPS step-down converter clock source used is indicated by the SMPSSWS in *RCC SMPS step-down converter control register (RCC\_SMPSCR)*.

A switch from one clock source to another only occurs if the target clock source is ready (clock stable after startup delay). If a clock source which is not yet ready is selected, the switch will occur when the clock source becomes ready. Status bits in the *RCC internal clock sources calibration register (RCC\_ICSCR)* indicate which clock(s) is (are) ready and the SMPSSWS bit indicates which clock is currently used as a SMPS step-down converter clock.

The division factor in SMPSDIV is dependent on the selected SMPS step-down converter clock source in SMPSEL. When MSI is selected in SMPSEL, The MSIRANGE is also taken into account to determine the selected MSI frequency. This allows to switching from one clock source to an other by keeping the same SMPS step-down converter clock frequency. The MSI shall only be selected as SMPS step-down converter clock source when a supported SMPS step-down converter clock MSIRANGE is set.

When the RF system is active HW will select automatically the HSE source with the current SMPSDIV division ratio.

**Table 39. SMPS step-down converter clock source selection and division**

SMPSSEL	HSI16	MSI				HSE
SMPSDIV	16 MHz	16 MHz	24 MHz	32 MHz	48 MHz	32 MHz
00	1 (8 MHz)	1 (8 MHz)	Reserved	2 (8 MHz)	3 (8 MHz)	2 (8 MHz)
01	2 (4 MHz)	2 (4 MHz)	3 (4 MHz)	4 (4 MHz)	6 (4 MHz)	4 (4 MHz)
10	Reserved					
11	Reserved					

### 8.2.15 ADC clock

The ADC clock is derived from the system clock, or from the PLL or PLLSAI1 output. It can reach 64 MHz and can be divided by the prescaler values 1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128 or 256 by configuring the ADC1\_CCR register. It is asynchronous to the AHB clock. Alternatively, the ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). This programmable factor is configured using the CKMODE bit fields in the ADC1\_CCR.

If the programmed factor is '1', the AHB prescaler must be set to '1'.

### 8.2.16 RTC clock

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

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

- If LSE is selected as RTC clock the RTC continues to work even if the  $V_{DD}$  supply is switched off, provided the  $V_{BAT}$  supply is maintained.
- If LSI is selected as the RTC clock the RTC state is not guaranteed if the  $V_{DD}$  supply is powered off.
- If the HSE clock divided by a prescaler is used as the RTC clock the RTC state is not guaranteed if the  $V_{DD}$  supply is powered off or if the internal voltage regulator is powered off (removing power from the  $V_{CORE}$  domain).

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

### 8.2.17 Timer clock

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

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

### 8.2.18 Watchdog clock

If the Independent watchdog (IWDG) is started by either hardware option or software access, the LSI clock is forced on.

When neither the LSI1 oscillator nor the LSI2 oscillator is enabled when starting the IWDG, the LSI1 oscillator is forced on. After the LSI1 oscillator temporization, the clock is provided to the IWDG.

When LSI2 is switched on with the LSI2ON bit, the LSI clock is switched from LSI1 to LSI2.

When LSI2 is disabled by LSI2ON (when the IWDG is running), the LSI1 oscillator is forced on. The LSI clock is switched to LSI1 when ready, and only then LSI2 is stopped.

### 8.2.19 True RNG clock

The true random number generator (RNG) seed clock is derived from the same clock selected for the USB, LSI or LSE clock.

### 8.2.20 Clock-out capability

- MCO

The microcontroller clock output (MCO) capability enables the clock to be output on the external MCO pin. One of the following clock signals can be selected as MCO clock:

- LSI1 (available in Run and Stop modes)
- LSI2 (available in Run and Stop modes)
- LSE (available in Run and Stop modes)
- MSI (available in Run mode)
- HSI (available in Run mode, when enabled by HSION)
- HSI48 (available in Run mode)
- HSE (available in Run mode)
- PLLRCLK (available in Run mode)
- SYSCLK (available in Run mode)

The selection is controlled by the MCOSEL[3:0] bits of the [RCC clock configuration register \(RCC\\_CFGR\)](#). The selected clock can be divided with the MCOPRE[2:0] field of the [RCC clock configuration register \(RCC\\_CFGR\)](#).

The clock on MCO is only available in Run and Stop modes and is not available in Standby and Shutdown modes.

- LSCO

The LSCO output enables a low speed clock to be output on the external LSCO pin:

- LSI (LSI1 or LSI2)
- LSE

The selection is controlled by the LSCOSEL, and enabled with the LSCOEN in the [RCC backup domain control register \(RCC\\_BDCR\)](#).

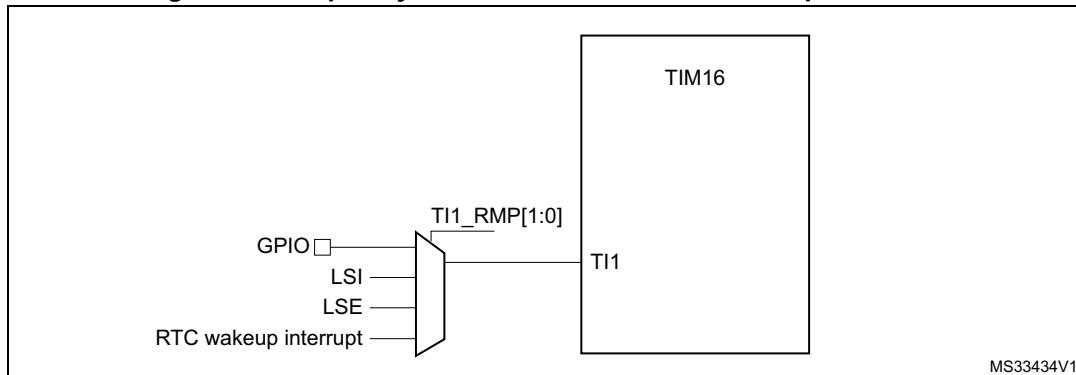
The clock on LSCO is available in Run, Stop, and on one GPIO in Standby and Shutdown modes.

The configuration registers of the corresponding GPIO port must be programmed in alternate function mode.

### 8.2.21 Internal/external clock measurement with TIM16/TIM17

It is possible to indirectly measure the frequency of all on-board clock sources by mean of the TIM16 or TIM17 channel 1 input capture, as shown on [Figure 18](#) and [Figure 19](#).

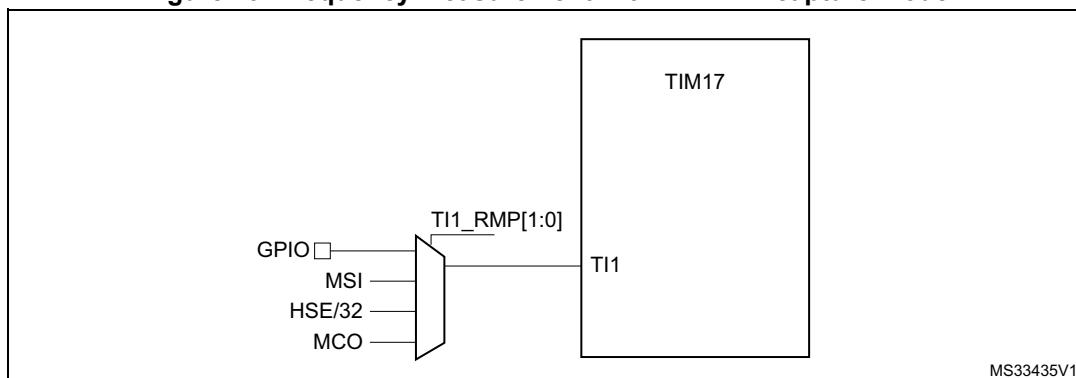
**Figure 18. Frequency measurement with TIM16 in capture mode**



The input capture channel of TIM16 can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1\_RMP[1:0] bits in the TIM16\_OR register. The possibilities are the following ones:

- TIM16 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.
- TIM16 Channel1 is connected to the LSI clock.
- TIM16 Channel1 is connected to the LSE clock.
- TIM16 Channel1 is connected to the RTC wakeup interrupt signal. In this case the RTC interrupt should be enabled.

**Figure 19. Frequency measurement with TIM17 in capture mode**



The input capture channel of the Timer 17 can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1\_RMP[1:0] bits in the TIM17\_OR register. The possibilities are the following ones:

- TIM17 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheet.
- TIM17 Channel1 is connected to the MSI clock.
- TIM17 Channel1 is connected to the HSE/32 clock.
- TIM17 Channel1 is connected to the microcontroller clock output (MCO), this selection is controlled by the MCOSEL[3:0] bits of the Clock configuration register (RCC\_CFRG).

### Calibration of the HSI16 and the MSI

For TIM16, the primary purpose of connecting the LSE to the channel 1 input capture is to be able to precisely measure the HSI16 and MSI system clocks (for this, either the HSI16 or MSI should be used as the system clock source). The number of HSI16 (MSI, respectively) clock counts between consecutive edges of the LSE signal provides a measure of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few ppm tens), it is possible to determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process- and/or temperature- and voltage-related frequency deviations.

Both the MSI and HSI16 oscillators have dedicated user-accessible calibration bits for this purpose.

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

If LSE is not available, HSE32 is the better option to reach the most precise possible calibration.

It is however impossible to have a good enough resolution when the MSI clock is low (typically below 1 MHz). In this case, it is advised to:

- accumulate the results of several captures in a row
- use the timer input capture prescaler (up to one capture every eight periods)
- use the RTC wakeup interrupt signal (when the RTC is clocked by the LSE) as the input for the channel1 input capture. This improves the measurement precision. For this purpose the RTC wakeup interrupt must be enable.

### Calibration of the LSI

The calibration of the LSI will follow the same pattern that for the HSI, but changing the reference clock. It is necessary to connect LSI clock to the channel 1 input capture of the TIM16. Then define the HSE as system clock source, the number of his clock counts between consecutive edges of the LSI signal provides a measure of the internal low speed clock period.

The basic concept consists in providing a relative measurement (e.g. the HSE/LSI ratio). The precision is therefore closely related to the ratio between the two clock sources, higher ratios result in more accurate measurements.

## 8.2.22 Peripheral clocks enable

Most peripheral bus and kernel clocks can individually be enabled per CPU. The RCC\_AHBxENR, and RCC\_APBxENR enable peripheral clocks for CPU1 and RCC\_C2\_AHBxENR, and RCC\_C2\_APBxENR for CPU2. The peripheral clocks will follow the CPU(s) state for which they are enabled, see [Table 40](#).

Peripheral bus clock activity during the CPU Sleep mode is controlled by the peripheral clock CPU sleep mode enable bit of the RCC\_AHBxSMENR, and RCC\_APBxSMENR for CPU1 and RCC\_C2\_AHBxSMENR, and RCC\_C2\_APBxSMENR for CPU2 registers. The peripheral bus clock during Sleep mode will follow the CPU(s) state for which is it enabled, see [Table 40](#).

Table 40. Peripheral clock enable

Peripheral					
xxxEN	xxxSMEN	CPU mode	System mode	Bus clock	Kernel clock <sup>(1)</sup>
0	x	Any	Any	Stopped	Stopped
		CRun	Run	Clocked	Clocked
		0	CSleep and CStop	Run	Stopped
			CSleep	Run	Clocked
		1	CStop	Run	Stopped
				Stop	Clocked when selected from – HSI16 – LSI – LSE.
1			Standby or Shutdown		Stopped when selected from – bus clock – SYSCLK – PLL clocks – MSI – HSI48 – PLLSAI1 clocks

1. Only the peripherals SAI, I2C, LPTIM, USART, LPUART, USB FS, true RNG and ADC have a kernel clock.

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

When the peripheral bus clock is active the peripheral can be accessed by both CPUs regardless of which CPU has enabled the peripheral bus clock in its CPU xxxEN bit. However, when the peripheral bus clock is enabled by only one CPU, and this CPU enters low power mode, the peripheral bus clock is stopped (also depending on this CPUs xxxSMEN setting) and peripheral access for the other CPU is no longer supported. It is therefore good practice to enable the peripheral bus clock with the CPU dedicated clock enable.

When the peripheral kernel clock is not active, the peripheral functionality is stopped.

The enable bit has a synchronization mechanism to create a glitch free clock for the peripheral. After the enable bit is set, there is a two clock cycles delay before the clock becomes active.

**Caution:** Just after enabling the clock for a peripheral, software must wait for a delay before accessing the peripheral registers.

**Note:** *The BLE-IP when active, will activate the BLE bus and SRAM2 bus interface clocks.*

## 8.3 Low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep and Low-power sleep modes stop the CPU clock. The memory interface clocks (Flash and SRAM1 and SRAM2 interfaces) can be stopped by software during Sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.
- Stop modes (Stop 0, Stop 1 and Stop 2) stops most clocks in the  $V_{CORE}$  domain and disables the PLLs, the MSI and the HSE oscillators. The HSI16 may be kept running when requested by the IPs (USART1, LPUART1, I2C1, I2C3) that make it possible to wakeup from Stop modes.

All U(S)ARTs, LPUARTs and I<sup>2</sup>Cs have the capability to enable the HSI16 oscillator even when the MCU is in Stop mode (if HSI16 is selected as the clock source for that peripheral).

All U(S)ARTs, LPUARTs and LPTIMs can also be driven by the LSE oscillator when the system is in Stop mode (if LSE is selected as clock source for that peripheral) and the LSE oscillator is enabled (LSEON). In that case the LSE remains always ON in Stop mode (they do not have the capability to turn on the LSE oscillator).

The LPTIMs can also be driven by the LSI oscillator when the system is in Stop mode (if LSI is selected as clock source for that peripheral) and the LSI oscillator is enabled (LSI1ON or LSI2ON).

- Standby and Shutdown modes stops all the clocks in the  $V_{CORE}$  domain and disables the PLL, the HSI, the MSI and the HSE oscillators.

The low power modes mode can be overridden for debugging by setting the DBG\_SLEEP, DBG\_STOP or DBG\_STANDBY bits in the DBGMCU\_CR register. In addition the EXTI CDBGWRUPREQ events can be used to allow debugging in Stop modes (see [Table 41](#)).

**Table 41. Single core Low power debug configurations<sup>(1)</sup>**

Mode	CDBGWRUPREQ		DBGMCU			Debug
	CPU1	DBG_STANDBY	DBG_STOP	DBG_SLEEP	CPU1	
Sleep	X	X	X	X		Enabled
Stop0 and Stop1	Disabled	X	Disabled	-		Disabled
	Enabled					Enabled
Stop0, Stop1 and Stop2	X		Enabled			Enabled
Standby	X	Disabled	-	-		Disabled
		Enabled				Enabled

1. X = Do not use.

When leaving the Stop modes (Stop0, Stop1 or Stop2), the system clock is either MSI or HSI, depending on the software configuration of the STOPWUCK bit in the RCC\_CFGR register. When the RF system is enabled the HSI16 shall be selected as in STOPWUCK. When STOPWUCK select the HSI16 clock when leaving Stop mode, the C2HPRE is reset to select divide by 1. The frequency (range and user trim) of the MSI oscillator is the one configured before entering Stop mode. The user trim of HSI16 is kept. If the MSI was in PLL-mode before entering Stop mode, the PLL-mode stabilization time must be waited for after

wakeup even if the LSE was kept ON during the Stop mode. Before entering Stop 0 mode with SMPS step-down converter enabled the HSI16 must be kept on by setting the HSIKERON register bit.

When leaving the Standby mode, the system clock is HSI.

When leaving the Shutdown modes, the system clock is MSI. The MSI frequency at wakeup from Shutdown mode is 4 MHz. The user trim is lost.

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

## 8.4 RCC registers

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

Address offset: 0x000

Reset value: 0x0000 0061 (after POR reset), 0x0000 0160 (after wakeup from Standby reset)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PLL SAI1RDY	PLL SAI1ON	PLL RDY	PLL ON	Res.	Res.	Res.	HSEPRE	CSS ON	Res.	HSE RDY	HSE ON
				r	rw	r	rw				rw	rs		r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	HSIKE RDY	HSI ASFS	HSI RDY	HSIKER ON	HSI ON	MSIRANGE[3:0]				Res.	MSI PLLEN	MSI RDY	MSI ON
			r	rw	r	rw	rw	rw	rw	rw	rw		rw	r	rw

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

Bit 27 **PLLSAI1RDY**: SAI PLL clock ready flag

Set by hardware to indicate that the PLLSAI1 is locked.

- 0: PLLSAI1 unlocked
- 1: PLLSAI1 locked

Bit 26 **PLLSAI1ON**: SAI PLL enable

Set and cleared by software to enable PLLSAI1.

Cleared by hardware when entering Stop, Standby or Shutdown mode.

- 0: PLLSAI1 OFF
- 1: PLLSAI1 ON

Bit 25 **PLLRDY**: System PLL clock ready flag

Set by hardware to indicate that the system PLL is locked.

- 0: PLL unlocked
- 1: PLL locked

Bit 24 **PLLON**: System PLL enable

Set and cleared by software to enable the PLL.

Cleared by hardware when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the PLL clock is used as the system clock.

- 0: PLL OFF
- 1: PLL ON

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

Bit 20 **HSEPRE**: HSE system clock and PLL M divider prescaler

Set and cleared by software to control the division factor of the system clock and PLL M divider input when selecting HSE clock.

- 0: SYSCLK and PLL M divider input clocks are not divided (HSE)
- 1: SYSCLK and PLL M divider input clocks are divided by 2 (HSE/2)

Bit 19 **CSSON**: HSE clock security system enable

Set by software to enable the clock security system. When CSSON is set, the HSE lock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if a HSE clock failure is detected. This bit is set only and is cleared by reset.

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

## Bit 18 Reserved, must be kept at reset value.

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

Set by hardware to indicate that the HSE oscillator is stable.

0: HSE oscillator not ready  
1: HSE oscillator ready

*Note: Once the HSEON bit is cleared, HSERDY goes low after six HSE clock cycles. Regardless of the value of the HSEON bit, the HSE oscillator starts on RF demand when needed. It is possible to have HSEON = 0 and HSERDY = 1.*

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.

0: HSE oscillator OFF  
1: HSE oscillator ON

## Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **HSIKERDY**: HSI16 kernel clock ready flag for peripherals requests.

Set by hardware to indicate that HSI16 oscillator is stable when enabled by HSIKERON or a peripheral kernel clock request. Not set when HSI16 is enabled by software by setting HSION, or by wakeup from Standby.

0: HSI16 oscillator not ready  
1: HSI16 oscillator ready

Bit 11 **HSIASFS**: HSI16 automatic start from Stop

Set and cleared by software. When the system wakeup clock is MSI, this bit is used to wakeup the HSI16 is parallel of the system wakeup.

0: HSI16 oscillator is not enabled by hardware when exiting Stop mode with MSI as wakeup clock.  
1: HSI16 oscillator is enabled by hardware when exiting Stop mode with MSI as wakeup clock.

Bit 10 **HSIRDY**: HSI16 clock ready flag. (After wakeup from Standby this bit is read 'b1 once the HSI16 is ready)

Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSI16 is enabled by software by setting HSION, or by wakeup from Standby. Not set when HSI16 is enabled by HSIKERON or by IP request.

0: HSI16 oscillator not ready  
1: HSI16 oscillator ready

*Note: Once the HSION bit is cleared, HSIRDY goes low after 6 HSI16 clock cycles.*

Bit 9 **HSIKERON**: HSI16 always enable for peripheral kernel clocks.

Set and cleared by software to force HSI16 ON even in Stop modes. The HSI16 enabled by HSIKERON can only feed USARTs, LPUARTs and I<sup>2</sup>Cs peripherals configured with HSI16 as kernel clock. Keeping the HSI16 ON in Stop mode avoids slowing down the communication speed because of the HSI16 startup time. This bit has no effect on HSION value.

0: No effect on HSI16 oscillator.

1: HSI16 oscillator is forced ON even in Stop mode.

Bit 8 **HSION**: HSI16 clock enable

Set and cleared by software.

Cleared by hardware to stop the HSI16 oscillator when entering Stop, Standby or Shutdown mode.

Set by hardware to force the HSI16 oscillator ON when STOPWUCK = 1 or HSIASFS = 1 when leaving Stop modes, or in case of failure of the HSE crystal oscillator.

This bit is set by hardware if the HSI16 is used directly or indirectly as system clock.

0: HSI16 oscillator OFF

1: HSI16 oscillator ON

Bits 7:4 **MSIRANGE[3:0]**: MSI clock ranges

These bits are configured by software to choose the frequency range of MSI when MSIRANGE is set. Twelve frequency ranges are available:

0000: range 0, ~100 kHz

0001: range 1, ~200 kHz

0010: range 2, ~400 kHz

0011: range 3, ~800 kHz

0100: range 4, ~1M Hz

0101: range 5, ~2 MHz

0110: range 6, ~4 MHz (reset value)

0111: range 7, ~8 MHz

1000: range 8, ~16 MHz

1001: range 9, ~24 MHz

1010: range 10, ~32 MHz

1011: range 11, ~48 MHz

others: not allowed (hardware write protection)

*Note: Warning: MSIRANGE can be modified when MSI is OFF (MSION = 0) or when MSI is ready (MSIRDY = 1). MSIRANGE must NOT be modified when MSI is ON and NOT ready (MSION = 1 and MSIRDY = 0).*

Bit 3 Reserved, must be kept at reset value.

Bit 2 **MSIPLLLEN**: MSI clock PLL enable

Set and cleared by software to enable/ disable the PLL part of the MSI clock source.

MSIPLLLEN must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware). There is a hardware protection to avoid enabling MSIPLLLEN if LSE is not ready.

This bit is cleared by hardware when LSE is disabled (LSEON = 0) or when the clock security system on LSE detects a LSE failure (refer to RCC\_CSR register).

0: MSI PLL OFF

1: MSI PLL ON

Bit 1 **MSIRDY**: MSI clock ready flag (After reset this bit is read 'b1 once the MSI is ready)

This bit is set by hardware to indicate that the MSI oscillator is stable.

0: MSI oscillator not ready

1: MSI oscillator ready

*Note: Once the MSION bit is cleared, MSIRDY goes low after 6 MSI clock cycles.*

Bit 0 **MSION: MSI clock enable**

This bit is set and cleared by software.

Cleared by hardware to stop the MSI oscillator when entering Stop, Standby or Shutdown mode.

Set by hardware to force the MSI oscillator ON when exiting Standby or Shutdown mode.

Set by hardware to force the MSI oscillator ON when STOPWUCK = 0 when exiting from Stop modes, or in case of a failure of the HSE oscillator

Set by hardware when used directly or indirectly as system clock.

0: MSI oscillator OFF

1: MSI oscillator ON

## 8.4.2 RCC internal clock sources calibration register (RCC\_ICSCR)

Address offset: 0x004

Reset value: 0x40XX 00XX (X is factory-programmed)

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

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

Bit 31 Reserved, must be kept at reset value.

### Bits 30:24 **HSITRIM[6:0]: HSI16 clock trimming**

These bits provide an additional user-programmable trimming value that is added to the HSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI16.

The default value is 64, which, when added to the HSICAL value, should trim the HSI16 to 16 MHz  $\pm 1\%$ .

### Bits 23:16 **HSICAL[7:0]: HSI16 clock calibration**

These bits are initialized at startup with the factory-programmed HSI16 calibration trim value. When HSITRIM is written, HSICAL is updated with the sum of HSITRIM and the factory trim value.

### Bits 15:8 **MSITRIM[7:0]: MSI clock trimming**

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

The default value is 0, which, when added to the MSICAL value, should trim the MSI to its mid frequency.

### Bits 7:0 **MSICAL[7:0]: MSI clock calibration**

These bits are initialized at startup with the factory-programmed MSI calibration trim value. When MSITRIM is written, MSICAL is updated with the sum of MSITRIM and the factory trim value.

### 8.4.3 RCC clock configuration register (RCC\_CFGR)

Address offset: 0x008

Reset value: 0x0007 0000 (after POR reset), 0x0007 0001 (after wakeup from Standby)

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

One or two wait states inserted only if the access occurs during clock source switch.

From 0 to 15 wait states inserted if the access occurs when the APB or AHB prescalers values update is on going.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCOPRE[2:0]			MCOSEL[3:0]				Res.	Res.	Res.	Res.	Res.	PPRE2F	PPRE1F	HPREF
	rw	rw	rw	rw	rw	rw	rw						r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP WUCK	Res.	PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **MCOPRE[2:0]**: Microcontroller clock output prescaler

These bits are set and cleared by software.

It is highly recommended to change this prescaler before MCO output is enabled.

000: MCO is divided by 1

001: MCO is divided by 2

010: MCO is divided by 4

011: MCO is divided by 8

100: MCO is divided by 16

Others: not allowed

Bits 27:24 **MCOSEL[3:0]**: Microcontroller clock output

Set and cleared by software.

0000: MCO output disabled, no clock on MCO

0001: SYSCLK system clock selected

0010: MSI clock selected.

0011: HSI16 clock selected.

0100: HSE clock selected (after stabilization, after HSERDY = 1)

0101: Main PLLRCLK clock selected

0110: LSI1 clock selected

0111: LSI2 clock selected

1000: LSE clock selected

1001: Internal HSI48 clock selected

1100: HSE clock selected (before stabilization, after HSEON = 1)

Others: Reserved

*Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.*

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

Bit 18 **PPRE2F**: PCLK2 prescaler flag (APB2)

Set and reset by hardware to acknowledge PCLK2 prescaler programming

Reset when a new prescaler value is programmed in PPRE2. set when the programmed value is actually applied.

0: PCLK2 prescaler value not yet applied

1: PCLK2 prescaler value applied

Bit 17 **PPRE1F**: PCLK1 prescaler flag (APB1)

Set and reset by hardware to acknowledge PCLK1 prescaler programming

Reset when a new prescaler value is programmed in PPRE1. set when the programmed value is actually applied.

0: PCLK1 prescaler value not yet applied

1: PCLK1 prescaler value applied

Bit 16 **HPREF**: HCLK1 prescaler flag (CPU1, AHB1, AHB2, AHB3 and SRAM1)

Set and reset by hardware to acknowledge HCLK1 prescaler programming

Reset when a new prescaler value is programmed in HPRE. set when the programmed value is actually applied.

0: HCLK1 prescaler value not yet applied

1: HCLK1 prescaler value applied

Bit 15 **STOPWUCK**: Wakeup from Stop and CSS backup clock selection

Set and cleared by software to select the system clock used when exiting Stop mode.

The selected clock is also used as emergency clock for the clock security system on HSE.

Warning: STOPWUCK must not be modified when the HSE clock security system is enabled by CSSON in [RCC clock control register \(RCC\\_CR\)](#) register and the system clock is HSE (SWS = 10) or a switch on HSE is requested (SW= 10).

0: MSI oscillator selected as wakeup from stop clock and CSS backup clock.

1: HSI16 oscillator selected as wakeup from stop clock and CSS backup clock

## Bit 14 Reserved, must be kept at reset value.

Bits 13:11 **PPRE2[2:0]**: PCLK2 high-speed prescaler (APB2)

Set and cleared by software to control the division factor of the PCLK2 clock (APB2).

The PPRE2F flag can be checked to know if the programmed PPRE2 prescaler value is applied.

0xx: HCLK1 not divided

100: HCLK1 divided by 2

101: HCLK1 divided by 4

110: HCLK1 divided by 8

111: HCLK1 divided by 16

Bits 10:8 **PPRE1[2:0]**: PCLK1 low-speed prescaler (APB1)

Set and cleared by software to control the division factor of the PCLK1 clock (APB1).

The PPRE1F flag can be checked to know if the programmed PPRE1 prescaler value is applied.

0xx: HCLK1 not divided

100: HCLK1 divided by 2

101: HCLK1 divided by 4

110: HCLK1 divided by 8

111: HCLK1 divided by 16

Bits 7:4 **HPRE[3:0]:** HCLK1 prescaler (CPU1, AHB1, AHB2, AHB3 and SRAM1.)

Set and cleared by software to control the division factor of the HCLK1 clock (CPU1, AHB1, AHB2, AHB3 and SRAM1).

The HPREF flag can be checked to know if the programmed HPRE prescaler value is applied.

**Caution:** Depending on the device voltage range, the software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details refer to [Section 6.1.6: Dynamic voltage scaling management](#)). After a write operation to these bits and before decreasing the voltage range the register bit HPREF must be read to be sure that the new value has been taken into account.

0001: SYSCLK divided by 3  
 0010: SYSCLK divided by 5  
 0101: SYSCLK divided by 6  
 0110: SYSCLK divided by 10  
 0111: SYSCLK divided by 32  
 1000: SYSCLK divided by 2  
 1001: SYSCLK divided by 4  
 1010: SYSCLK divided by 8  
 1011: SYSCLK divided by 16  
 1100: SYSCLK divided by 64  
 1101: SYSCLK divided by 128  
 1110: SYSCLK divided by 256  
 1111: SYSCLK divided by 512  
 Others: SYSCLK not divided

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

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

00: MSI oscillator used as system clock  
 01: HSI16 oscillator used as system clock  
 10: HSE used as system clock  
 11: PLL used as system clock

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

Set and cleared by software to select system clock source (SYSCLK).

Configured by HW to force MSI oscillator selection when exiting Shutdown mode.

Configured by HW to force HSI16 oscillator selection when exiting Standby mode.

Configured by HW to force MSI or HSI16 oscillator selection when exiting Stop mode or in case of failure of the HSE oscillator, depending on STOPWUCK value.

00: MSI selected as system clock  
 01: HSI16 selected as system clock  
 10: HSE selected as system clock  
 11: PLL selected as system clock

**Note:** The latency when changing SW and SWS bits is set according to the new configuration when switching the SYSCLK clock source, in a maximum of three cycles of the previous clock.

#### 8.4.4 RCC PLL configuration register (RCC\_PLLCFGR)

Address offset: 0x00C

Reset value: 0x2204 0100

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

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

- $f(\text{VCO clock}) = f(\text{PLL clock input}) \times (\text{PLL}N / \text{PLL}M)$
- $f(\text{PLL\_P}) = f(\text{VCO clock}) / \text{PLL}P$
- $f(\text{PLL\_Q}) = f(\text{VCO clock}) / \text{PLL}Q$
- $f(\text{PLL\_R}) = f(\text{VCO clock}) / \text{PLL}R$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLL[R:0]			PLLREN		PLLQ[R:0]			PLLQEN	Res.	Res.	PLL[P:0]				PLL[PEN]
rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PLL[N:0]							Res.	PLL[M:0]			Res.	Res.	PLL[SR:0]	
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:29 **PLL[R:0]**: Main PLL division factor for PLLRCLK

Set and cleared by software to control the frequency of the main PLL output clock PLLRCLK. This output can be selected as system clock. These bits can be written only if the PLL is disabled.

PLLCLK output clock frequency = VCO frequency / PLLR with PLLR = 2, 3, 4,... or 8  
[VCO frequency / (N + 1)]

000: reserved

001: PLLR = 2

010: PLLR = 3

011: PLLR = 4

100: PLLR = 5

101: PLLR = 6

110: PLLR = 7

111: PLLR = 8

The software has to set these bits so that 64 MHz is not exceeded on this domain.

Bit 28 **PLLREN**: Main PLL PLLRCLK output enable

Set and reset by software to enable the PLLRCLK output of the main PLL (used as system clock).

This bit cannot be written when PLLRCLK output of the PLL is used as System clock.

To save power, when the PLLRCLK output of the PLL is not used, the value of PLLREN should be 0.

0: PLLRCLK output disabled

1: PLLRCLK output enabled

Bits 27:25 **PLLQ[2:0]:** Main PLL division factor for PLLQCLK

Set and cleared by software to control the frequency of the main PLL output clock PLLQCLK. This output can be selected for USB and true RNG clock. These bits can be written only if PLL is disabled.

PLLQCLK output clock frequency = VCO frequency / PLLQ with PLLQ = 2, 3, 4,... or 8 [VCO frequency / (N + 1)]

000: reserved

001: PLLQ = 2

010: PLLQ = 3

011: PLLQ = 4

100: PLLQ = 5

101: PLLQ = 6

110: PLLQ = 7

111: PLLQ = 8

The software has to set these bits so that 64 MHz is not exceeded on this domain.

Bit 24 **PLLQEN:** Main PLL PLLQCLK output enable

Set and reset by software to enable the PLLQCLK output of the main PLL (used as system clock).

In order to save power, when the PLLQCLK output of the PLL is not used, the value of PLLQEN should be 0.

0: PLLQCLK output disable

1: PLLQCLK output enable

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:17 **PLLP[4:0]:** Main PLL division factor for PLLPCLK.

Set and cleared by software to control the frequency of the main PLL output clock PLLPCLK. This output can be selected for SAI1 and ADC. These bits can be written only if PLL is disabled.

PLLPCLK output clock frequency = VCO frequency / PLLP with PLLP = 2, 3, 4,... or 32 [VCO frequency / (N + 1)]

0000: reserved

00001: PLLP = 2

00010: PLLP = 3

00011: PLLP = 4

00100: PLLP = 5

...

11111: PLLP = 32

The software has to set these bits so that 64 MHz is not exceeded on this domain.

Bit 16 **PLLPEN:** Main PLL PLLPCLK output enable

Set and reset by software to enable the PLLPCLK output of the main PLL.

In order to save power, when the PLLPCLK output of the PLL is not used, the value of PLLPEN should be 0.

0: PLLPCLK output disable

1: PLLPCLK output enable

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLLN[6:0]**: Main 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 the PLL is disabled.

VCO output frequency = VCO input frequency  $\times$  PLLN with  $6 \leq \text{PLLN} \leq 127$

0000000: PLLN = 0 reserved (must not be used)

0000001: PLLN = 1 reserved (must not be used)

...

0000101: PLLN = 5 reserved (must not be used)

0000110: PLLN = 6

1111111: PLLN = 127

The software has to set these bits to ensure that the VCO output frequency is between 96 and 344 MHz.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PLLM[2:0]**: Division factor for the main PLL and audio PLLSAI1 input clock

Set and cleared by software to divide the PLL and PLLSAI1 input clock before the VCO. These bits can be written only when all PLLs are disabled.

VCO input frequency = PLL input clock frequency / PLLM with  $1 \leq \text{PLLM} \leq 8$

000: PLLM = 1

001: PLLM = 2

010: PLLM = 3

011: PLLM = 4

100: PLLM = 5

101: PLLM = 6

110: PLLM = 7

111: PLLM = 8

The software has to set these bits to ensure that the VCO input frequency ranges from 2.66 to 16 MHz.

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

Bits 1:0 **PLLSRC[1:0]**: Main PLL and audio PLLSAI1 entry clock source

Set and cleared by software to select PLL and PLLSAI1 clock source. These bits can be written only when PLL and PLLSAI1 are disabled.

In order to save power, when no PLL is used, the value of PLLSRC should be 00.

00: No clock sent to PLL and PLLSAI1

01: MSI clock selected as PLL and PLLSAI1 clock entry

10: HSI16 clock selected as PLL and PLLSAI1 clock entry

11: HSE clock selected as PLL and PLLSAI1 clock entry

### 8.4.5 RCC PLLSAI1 configuration register (RCC\_PLLSAI1CFGR)

Address offset: 0x010

Reset value: 0x2204 0100

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

This register is used to configure the PLLSAI1 clock outputs according to the formulas:

- $f(\text{VCOSAI clock}) = f(\text{PLLAI1 clock input}) \times (\text{PLL N} / \text{PLL M})$
- $f(\text{PLLAI1\_P}) = f(\text{VCOSAI clock}) / \text{PLL P}$
- $f(\text{PLLAI1\_Q}) = f(\text{VCOSAI clock}) / \text{PLL Q}$
- $f(\text{PLLAI1\_R}) = f(\text{VCOSAI clock}) / \text{PLL R}$

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

Bits 31:29 **PLL R[2:0]**: Audio PLLSAI1 division factor for PLLSAI1RCLK

Set and cleared by software to control the frequency of the audio PLLSAI1 output clock PLLSAI1RCLK. This output can be selected as system clock. These bits can be written only if PLLSAI1 is disabled.

PLLSAI1RCLK output clock frequency = VCO frequency / PLLR with PLLR = 2, 3, 4,... or 8 [VCO frequency / (N + 1)]

- 000: reserved
- 001: PLLR = 2
- 010: PLLR = 3
- 011: PLLR = 4
- 100: PLLR = 5
- 101: PLLR = 6
- 110: PLLR = 7
- 111: PLLR = 8

The software has to set these bits correctly not to exceed 64 MHz on this domain.

Bit 28 **PLLREN**: Audio PLLSAI1 PLLSAI1RCLK output enable

Set and reset by software to enable the PLLSAI1RCLK output of the audio PLLSAI1 (used as system clock).

In order to save power, when the PLLSAI1RCLK output of the PLLSAI1 is not used, the value of PLLREN should be 0.

- 0: PLLSAI1RCLK output disable
- 1: PLLSAI1RCLK output enable

Bits 27:25 **PLLQ[2:0]**: Audio PLLSAI1 division factor for PLLSAI1QCLK

Set and cleared by software to control the frequency of the audio PLLSAI1 output clock PLLSAI1QCLK. This output can be selected for USB and True RNG clock. These bits can be written only if PLLSAI1 is disabled.

PLLSAI1QCLK output clock frequency = VCO frequency / PLLQ with PLLQ = 2, 3, 4,... or 8  
[VCO frequency / (N + 1)]

000: reserved

001: PLLQ = 2

010: PLLQ = 3

011: PLLQ = 4

100: PLLQ = 5

101: PLLQ = 6

110: PLLQ = 7

111: PLLQ = 8

The software has to set these bits correctly not to exceed 64 MHz on this domain.

Bit 24 **PLLQEN**: Audio PLLSAI1 PLLSAI1QCLK output enable

Set and reset by software to enable the PLLSAI1QCLK output of the audio PLLSAI1 (used as system clock).

In order to save power, when the PLLSAI1QCLK output of the PLLSAI1 is not used, the value of PLLQEN should be 0.

0: PLLSAI1QCLK output disable

1: PLLSAI1QCLK output enable

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:17 **PLLP[4:0]**: Audio PLLSAI1 division factor for PLLSAI1PCLK.

Set and cleared by software to control the frequency of the audio PLLSAI1 output clock PLLSAI1PCLK. This output can be selected for SAI1 and ADC. These bits can be written only if PLLSAI1 is disabled.

PLLSAI1PCLK output clock frequency = VCO frequency / PLLP with PLLP = 2, 3, 4, ...or 32  
[VCO frequency / (N + 1)]

00000: reserved

00001: PLLP = 2

00010: PLLP = 3

00011: PLLP = 4

00100: PLLP = 5

...

11111: PLLP = 32

**Caution:** The software has to set these bits correctly not to exceed 64 MHz on this domain.

Bit 16 **PLLPEN**: Audio PLLSAI1 PLLSAI1PCLK output enable

Set and reset by software to enable the PLLSAI1PCLK output of the audio PLLSAI1.

In order to save power, when the PLLSAI1PCLK output of the PLL is not used, the value of PLLPEN should be 0.

0: PLLSAI1PCLK output disable

1: PLLSAI1PCLK output enable

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLLN[6:0]**: Audio PLLSAI1 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 PLLSAI1 is disabled.

VCO output frequency = VCO input frequency  $\times$  PLLN with  $4 \leq \text{PLLN} \leq 86$

0000000: PLLN = 0 wrong configuration

0000001: PLLN = 1 wrong configuration

...

0000011: PLLN = 3 wrong configuration

0000100: PLLN = 4

0000101: PLLN = 5

...

1010101: PLLN = 85

1010110: PLLN = 86

1010111: PLLN = 87 wrong configuration

...

1111111: PLLN = 127 wrong configuration

**Caution:** The software has to set correctly these bits to ensure that the VCO output frequency is between 64 and 344 MHz.

Bits 7:0 Reserved, must be kept at reset value.

#### 8.4.6 RCC clock interrupt enable register (RCC\_CIER)

Address offset: 0x018

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LSI2 RDYIE	HSI48 RDYIE	LSE CSSIE	Res.	Res.	PLLSAI1 RDYIE	PLL RDYIE	HSE RDYIE	HSI RDYIE	MSI RDYIE	LSE RDYIE	LSI1 RDYIE
				rw	rw	rw			rw	rw	rw	rw	rw	rw	rw

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

Bit 11 **LSI2RDYIE**: LSI2 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSI2 oscillator stabilization.

0: LSI2 ready interrupt disabled

1: LSI2 ready interrupt enabled

Bit 10 **HSI48RDYIE**: HSI48 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the internal HSI48 oscillator.

0: HSI48 ready interrupt disabled

1: HSI48 ready interrupt enabled

- Bit 9 **LSECSSIE:** LSE clock security system interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by the clock security system on LSE.  
 0: Clock security interrupt caused by LSE clock failure disabled  
 1: Clock security interrupt caused by LSE clock failure enabled
- Bits 8:7 Reserved, must be kept at reset value.
- Bit 6 **PLLSAI1RDYIE:** PLLSAI1 ready interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by PLLSAI1 lock.  
 0: PLLSAI1 lock interrupt disabled  
 1: PLLSAI1 lock interrupt enabled
- Bit 5 **PLLRDYIE:** PLL ready interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by PLL lock.  
 0: PLL lock interrupt disabled  
 1: PLL lock interrupt enabled
- Bit 4 **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 3 **HSIRDYIE:** HSI16 ready interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by the HSI16 oscillator stabilization.  
 0: HSI16 ready interrupt disabled  
 1: HSI16 ready interrupt enabled
- Bit 2 **MSIRDYIE:** MSI ready interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by the MSI oscillator stabilization.  
 0: MSI ready interrupt disabled  
 1: MSI ready interrupt enabled
- Bit 1 **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 0 **LSI1RDYIE:** LSI1 ready interrupt enable  
 Set and cleared by software to enable/disable interrupt caused by the LSI1 oscillator stabilization.  
 0: LSI1 ready interrupt disabled  
 1: LSI1 ready interrupt enabled

#### 8.4.7 RCC clock interrupt flag register (RCC\_CIFR)

Address offset: 0x01C

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LSI2 RDYF	HSI48R RDYF	LSE CSSF	CSSF	Res.	PLLSAI1 RDYF	PLL RDYF	HSE RDYF	HSI RDYF	MSI RDYF	LSE RDYF	LSI1 RDYF
				r	r	r	r		r	r	r	r	r	r	r

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

**Bit 11 LSI2RDYF:** LSI2 ready interrupt flag

Set by hardware when the LSI2 clock becomes stable and LSI2RDYDIE is set.

Cleared by software setting the LSI2RDYC bit.

0: No clock ready interrupt caused by the LSI2 oscillator

1: Clock ready interrupt caused by the LSI2 oscillator

**Bit 10 HSI48RDYF:** HSI48 ready interrupt flag

Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set in a response to setting the HSI48ON (refer to [RCC clock recovery RC register \(RCC\\_CRRCR\)](#)).

Cleared by software setting the HSI48RDYC bit.

0: No clock ready interrupt caused by the HSI48 oscillator

1: Clock ready interrupt caused by the HSI48 oscillator

**Bit 9 LSECSSF:** LSE clock security system interrupt flag

Set by hardware when a failure is detected in the LSE oscillator.

Cleared by software setting the LSECSSC bit.

0: No clock security interrupt caused by LSE clock failure

1: Clock security interrupt caused by LSE clock failure

**Bit 8 CSSF:** HSE clock security system interrupt flag

Set by hardware when a failure is detected in the HSE oscillator.

Cleared by software setting the CSSC bit.

0: No clock security interrupt caused by HSE clock failure

1: Clock security interrupt caused by HSE clock failure

Bit 7 Reserved, must be kept at reset value.

**Bit 6 PLLSAI1RDYF:** PLLSAI1 ready interrupt flag

Set by hardware when the PLLSAI1 locks and PLLSAI1RDYDIE is set.

Cleared by software setting the PLLSAI1RDYC bit.

0: No clock ready interrupt caused by PLLSAI1 lock

1: Clock ready interrupt caused by PLLSAI1 lock

**Bit 5 PLLRDYF:** PLL ready interrupt flag

Set by hardware when the PLL locks and PLLRDYDIE is set.

Cleared by software setting the PLLRDYC bit.

0: No clock ready interrupt caused by PLL lock

1: Clock ready interrupt caused by PLL lock

**Bit 4 HSERDYF:** HSE ready interrupt flag

Set by hardware when the HSE clock becomes stable and HSERDYDIE is set.

Cleared by software setting the HSERDYC bit.

0: No clock ready interrupt caused by the HSE oscillator

1: Clock ready interrupt caused by the HSE oscillator

Bit 3 **HSIRDYF**: HSI16 ready interrupt flag

Set by hardware when the HSI16 clock becomes stable and HSIRDYDIE is set in a response to setting the HSION (refer to *RCC clock control register (RCC\_CR)*). When HSION is not set but the HSI16 oscillator is enabled by the peripheral through a clock request, this bit is not set and no interrupt is generated.

Cleared by software setting the HSIRDYC bit.

0: No clock ready interrupt caused by the HSI16 oscillator

1: Clock ready interrupt caused by the HSI16 oscillator

Bit 2 **MSIRDYF**: MSI ready interrupt flag

Set by hardware when the MSI clock becomes stable and MSIRDYDIE is set.

Cleared by software setting the MSIRDYC bit.

0: No clock ready interrupt caused by the MSI oscillator

1: Clock ready interrupt caused by the MSI oscillator

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

Set by hardware when the LSE clock becomes stable and LSERDYDIE is set.

Cleared by software setting the LSERDYC bit.

0: No clock ready interrupt caused by the LSE oscillator

1: Clock ready interrupt caused by the LSE oscillator

Bit 0 **LSI1RDYF**: LSI1 ready interrupt flag

Set by hardware when the LSI1 clock becomes stable and LSI1RDYDIE is set.

Cleared by software setting the LSI1RDYC bit.

0: No clock ready interrupt caused by the LSI1 oscillator

1: Clock ready interrupt caused by the LSI1 oscillator

#### 8.4.8 RCC clock interrupt clear register (RCC\_CICR)

Address offset: 0x020

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LSI2 RDYC	HSI48 RDYC	LSE CSSC	CSSC	Res.	PLLSA1 RDYC	PLL RDYC	HSE RDYC	HSI RDYC	MSI RDYC	LSE RDYC	LSI1 RDYC

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

Bit 11 **LSI2RDYC**: LSI2 ready interrupt clear

This bit is set by software to clear the LSI2RDYF flag.

0: No effect

1: LSI2RDYF cleared

Bit 10 **HSI48RDYC**: HSI48 oscillator ready interrupt clear

This bit is set by software to clear the HSI48RDYF flag.

0: No effect

1: Clear the HSI48RDYC flag

- Bit 9 **LSECSSC**: LSE clock security system interrupt clear  
 This bit is set by software to clear the LSECSSF flag.  
 0: No effect  
 1: Clear LSECSSF flag
- Bit 8 **CSSC**: HSE clock security system interrupt clear  
 This bit is set by software to clear the HSE CSSF flag.  
 0: No effect  
 1: Clear HSE CSSF flag
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **PLLSAI1RDYC**: PLLSAI1 ready interrupt clear  
 This bit is set by software to clear the PLLSAI1RDYF flag.  
 0: No effect  
 1: Clear PLLSAI1RDYF flag
- Bit 5 **PLLRDYC**: PLL ready interrupt clear  
 This bit is set by software to clear the PLLRDYF flag.  
 0: No effect  
 1: Clear PLLRDYF flag
- Bit 4 **HSERDYC**: HSE ready interrupt clear  
 This bit is set by software to clear the HSERDYF flag.  
 0: No effect  
 1: Clear HSERDYF flag
- Bit 3 **HSIRDYC**: HSI16 ready interrupt clear  
 This bit is set software to clear the HSIRDYF flag.  
 0: No effect  
 1: Clear HSIRDYF flag
- Bit 2 **MSIRDYC**: MSI ready interrupt clear  
 This bit is set by software to clear the MSIRDYF flag.  
 0: No effect  
 1: MSIRDYF cleared
- Bit 1 **LSERDYC**: LSE ready interrupt clear  
 This bit is set by software to clear the LSERDYF flag.  
 0: No effect  
 1: LSERDYF cleared
- Bit 0 **LSI1RDYC**: LSI1 ready interrupt clear  
 This bit is set by software to clear the LSI1RDYF flag.  
 0: No effect  
 1: LSI1RDYF cleared

#### 8.4.9 RCC SMPS step-down converter control register (RCC\_SMPSCR)

Address offset: 0x024

Reset value: 0x0000 0301 (after POR reset), 0x0000 0300 (after wakeup from Standby)

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

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

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

Bits 9:8 **SMPSSWS[1:0]**: SMPS step-down converter clock switch status

Set and cleared by hardware to indicate which clock source is used as SMPS step-down converter clock when the SMPS is enabled. Whenever the HSE is active the HSE is used regardless of the settings in SMPSEL. Whenever the SMPS step-down converter is disabled in PWR\_CR5.SMPSEN, no clock is used.

00: HSI16 oscillator used as SMPS step-down converter clock

01: MSI oscillator used as SMPS step-down converter clock

10: HSE used as SMPS step-down converter clock

11: no clock is used

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

Bits 5:4 **SMPSDIV[1:0]**: SMPS step-down converter clock prescaler

Set and cleared by software to control the division factor of the SMPS step-down converter clock. The SMPS step-down converter clock prescaler factor depend on both the SMPSDIV[1:0] and SMPSEL[1:0] setting. See [Table 39](#).

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

Bits 1:0 **SMPSEL[1:0]**: SMPS step-down converter clock selection.

Set and cleared by software to select SMPS step-down converter clock source (SMPCLK).

00: HSI16 selected as SMPS step-down converter clock

01: MSI selected as SMPS step-down converter clock (the MSITRANGE shall be set to a supported value, see [Table 39](#)).

10: HSE selected as SMPS step-down converter clock

11: Reserved

**Note:** *On a POR reset or NRST pin reset or when waking up from Shutdown the SMPSEL is forced by hardware to select MSI (value b01). When waking up from Standby the SMPSEL is forced by hardware to select HSI (value b00). When waking up from Stop modes the SMPSEL is forced by hardware to select the MSI or HSI clock as defined in RCC\_CFG.RSTOPWUCK.*

#### 8.4.10 RCC AHB1 peripheral reset register (RCC\_AHB1RSTR)

Address offset: 0x028

Reset value: 0x00000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC RST
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRC RST	Res.	DMAMUX1 RST	DMA2 RST	DMA1 RST								
			rw										rw	rw	rw

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

Bit 16 **TSCRST**: Touch sensing controller reset (STM32WB55xx only)

Set and cleared by software.

0: No effect

1: Reset TSC

Note that this bit is reserved on STM32WB35xx.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC reset

Set and cleared by software.

0: No effect

1: Reset CRC

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1RST**: DMAMUX reset

Set and cleared by software.

0: No effect

1: Reset DMAMUX1

Bit 1 **DMA2RST**: DMA2 reset

Set and cleared by software.

0: No effect

1: Reset DMA2

Bit 0 **DMA1RST**: DMA1 reset

Set and cleared by software.

0: No effect

1: Reset DMA1

#### 8.4.11 RCC AHB2 peripheral reset register (RCC\_AHB2RSTR)

Address offset: 0x02C

Reset value: 0x000000000000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AES1 RST
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADC RST	Res.	Res.	Res.	Res.	Res.	GPIOH RST	Res.	Res.	GPIOE RST	GPIOD RST	GPIOC RST	GPIOB RST	GPIOA RST
		rw						rw			rw	rw	rw	rw	rw

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

Bit 16 **AES1RST**: AES1 hardware accelerator reset

Set and cleared by software.

0: No effect

1: Reset AES1

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **ADCRST**: ADC reset

Set and cleared by software.

0: No effect

1: Reset ADC interface

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHRST**: IO port H reset

Set and cleared by software.

0: No effect

1: Reset IO port H

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **GPIOERST**: IO port E reset

Set and cleared by software.

0: No effect

1: Reset IO port E

Bit 3 **GPIODRST**: IO port D reset (STM32WB55xx only)

Set and cleared by software.

0: No effect

1: Reset IO port D

Note that this bit is reserved on STM32WB35xx.

Bit 2 **GPIOCRST**: IO port C reset

Set and cleared by software.

0: No effect

1: Reset IO port C

Bit 1 **GPIOBRST**: IO port B reset

Set and cleared by software.

0: No effect

1: Reset IO port B

Bit 0 **GPIOARST**: IO port A reset

Set and cleared by software.

0: No effect

1: Reset IO port A

#### 8.4.12 RCC AHB3 and AHB4 peripheral reset register (RCC\_AHB3RSTR)

Address offset: 0x030

Reset value: 0x00000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLASH RST	Res.	Res.	Res.	Res.	IPCC RST	HSEM RST	RNG RST	AES2 RST	PKA RST
						rw					rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QUADSPI RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
							rw								

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHRST**: Flash interface reset

This bit can only be set when the Flash memory is in power-down. Set and cleared by software.

0: No effect

1: Reset Flash interface.

Bits 24:21 Reserved, must be kept at reset value.

Bit 20 **IPCCRST**: IPCC interface reset

Set and cleared by software.

0: No effect

1: Reset IPCC

Bit 19 **HSEMRST**: HSEM reset

Set and cleared by software.

0: No effect

1: Reset HSEM

Bit 18 **RNGRST**: True RNG reset

Set and cleared by software.

0: No effect

1: Reset true RNG

Bit 17 **AES2RST**: AES2 hardware accelerator reset

Set and cleared by software.

0: No effect

1: Reset AES2

Bit 16 **PKARST**: PKA hardware accelerator reset

Set and cleared by software.

0: No effect

1: Reset PKA

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **QUADSPIRST**: Quad SPI memory interface reset

Set and cleared by software.

0: No effect

1: Reset QUADSPI

Bits 7:0 Reserved, must be kept at reset value.

#### 8.4.13 RCC APB1 peripheral reset register 1 (RCC\_APB1RSTR1)

Address offset: 0x038

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
LPTIM1 RST	Res.	Res.	Res.	Res.	USB RST	Res.	.CRS RST	I2C3 RST	Res.	I2C1 RST	Res.	Res.	Res.	Res.	Res.
rw					rw		rw	rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2 RST	Res.	Res.	Res.	Res.	.LCD RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2 RST
	rw					rw									rw

Bit 31 **LPTIM1RST:** Low power timer 1 reset

Set and cleared by software.

0: No effect

1: Reset LPTIM1

Bits 30:27 Reserved, must be kept at reset value.

Bit 26 **USBRST:** USB FS reset

Set and cleared by software.

0: No effect

1: Reset the USB FS

Bit 25 Reserved, must be kept at reset value

Bit 24 **CRSRST:** CRS reset

Set and cleared by software.

0: No effect

1: Reset the CRS

Bit 23 **I2C3RST:** I2C3 reset

Set and reset by software.

0: No effect

1: Resets I2C3

Bit 22 Reserved, must be kept at reset value

Bit 21 **I2C1RST:** I2C1 reset

Set and cleared by software.

0: No effect

1: Reset I2C1

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **SPI2RST:** SPI2 reset (STM32WB55xx only)

Set and cleared by software.

0: No effect

1: Reset SPI2

Note that this bit is reserved on STM32WB35xx.

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

Bit 9 **LCDRST**: LCD interface reset (STM32WB55xx only)

Set and cleared by software.

0: No effect

1: Reset LCD

Note that this bit is reserved on STM32WB35xx.

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **TIM2RST**: TIM2 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM2

#### 8.4.14 RCC APB1 peripheral reset register 2 (RCC\_APB1RSTR2)

Address offset: 0x03C

Reset value: 0x0000000000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPTIM2 RST	Res.	Res.	Res.	Res.	LPUART1 RST									
										rw					rw

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

Bit 5 **LPTIM2RST**: Low-power timer 2 reset

Set and cleared by software.

0: No effect

1: Reset LPTIM2

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **LPUART1RST**: Low-power UART 1 reset

Set and cleared by software.

0: No effect

1: Reset LPUART1

#### 8.4.15 RCC APB2 peripheral reset register (RCC\_APB2RSTR)

Address offset: 0x040

Reset value: 0x0000000000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1 RST	Res.	Res.	TIM17 RST	TIM16 RST	Res.
										rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 RST	Res.	SPI1 RST	TIM1 RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **SAI1RST**: Serial audio interface 1 (SAI1) reset

Set and cleared by software.

0: No effect

1: Reset SAI1

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17RST**: TIM17 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM17 timer

Bit 17 **TIM16RST**: TIM16 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM16 timer

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1RST**: USART1 reset

Set and cleared by software.

0: No effect

1: Reset USART1

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1RST**: SPI1 reset

Set and cleared by software.

0: No effect

1: Reset SPI1

Bit 11 **TIM1RST**: TIM1 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM1 timer

Bits 10:0 Reserved, must be kept at reset value.

#### 8.4.16 RCC APB3 peripheral reset register (RCC\_APB3RSTR)

Address offset: 0x044

Reset value: 0x00000 0000

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

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

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

Bit 0 **RFRST**: Radio system BLE and 802.15.4 reset.

Set and cleared by software.

0: No effect

1: Reset radio system BLE and 802.15.4. The reset status of the radio system can be obtained from RFRSTS in [RCC control/status register \(RCC\\_CSR\)](#).

#### 8.4.17 RCC AHB1 peripheral clock enable register (RCC\_AHB1ENR)

Address offset: 0x048

Reset value: 0x0000 0000

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

**Note:** *When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCEN
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCEN	Res.	DMAMUX1EN	DMA2EN	DMA1EN								
			rw										rw	rw	rw

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

Bit 16 **TSCEN**: CPU1 Touch sensing controller clock enable (STM32WB55xx only)

Set and cleared by software.

0: TSC clock disable for CPU1

1: TSC clock enable for CPU1

Note that this bit is reserved on STM32WB35xx.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CPU1 CRC clock enable

Set and cleared by software.

0: CRC clock disable for CPU1

1: CRC clock enable for CPU1

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1**: CPU1 DMAMUX1 clock enable

Set and cleared by software.

0: DMAMUX1 clock disable for CPU1

1: DMAMUX1 clock enable for CPU1

Bit 1 **DMA2EN**: CPU1 DMA2 clock enable

Set and cleared by software.

0: DMA2 clock disable for CPU1

1: DMA2 clock enable for CPU1

Bit 0 **DMA1EN**: CPU1 DMA1 clock enable

Set and cleared by software.

0: DMA1 clock disable for CPU1

1: DMA1 clock enable for CPU1

#### 8.4.18 RCC AHB2 peripheral clock enable register (RCC\_AHB2ENR)

Address offset: 0x04C

Reset value: 0x0000 0000

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

**Note:** *When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AES1 EN
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADC EN	Res.	Res.	Res.	Res.	Res.	GPIOH EN	Res.	Res.	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
		rw						rw			rw	rw	rw	rw	rw

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

Bit 16 **AES1EN**: CPU1 AES1 accelerator clock enable

Set and cleared by software.

0: AES clock disabled for CPU1

1: AES clock enabled for CPU1

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **ADCEN**: CPU1 ADC clocks enable

Set and cleared by software.

0: ADC bus and kernel clocks disabled for CPU1

1: ADC bus and kernel clocks enabled for CPU1

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHEN**: CPU1 IO port H clock enable

Set and cleared by software.

0: IO port H clock disabled for CPU1

1: IO port H clock enabled for CPU1

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **GPIOEEN**: CPU1 IO port E clock enable

Set and cleared by software.

0: IO port E clock disabled for CPU1

1: IO port E clock enabled for CPU1

Bit 3 **GPIODEN**: CPU1 IO port D clock enable (STM32WB55xx only)

Set and cleared by software.

0: IO port D clock disabled for CPU1

1: IO port D clock enabled for CPU1

Note that this bit is reserved on STM32WB35xx.

Bit 2 **GPIOCEN**: CPU1 IO port C clock enable

Set and cleared by software.

0: IO port C clock disabled for CPU1

1: IO port C clock enabled for CPU1

Bit 1 **GPIOBEN**: CPU1 IO port B clock enable

Set and cleared by software.

0: IO port B clock disabled for CPU1

1: IO port B clock enabled for CPU1

Bit 0 **GPIOAEN**: CPU1 IO port A clock enable

Set and cleared by software.

0: IO port A clock disabled for CPU1

1: IO port A clock enabled for CPU1

#### 8.4.19 RCC AHB3 and AHB4 peripheral clock enable register (RCC\_AHB3ENR)

Address offset: 0x050

Reset value: 0x0208 0000

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

**Note:** When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLASH EN	Res.	Res.	Res.	Res.	IPCC EN	HSEM EN	RNG EN	AES2 EN	PKA EN
						rw					rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QUADSPI EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
							rw								

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHEN** CPU1 Flash memory interface clock enable

This bit can only be cleared when the Flash memory is in Power down. Set and cleared by software.

0: Flash interface clock disable for CPU1

1: Flash interface clock enable for CPU1

Bits 24:21 Reserved, must be kept at reset value.

Bit 20 **IPCCEN** CPU1 IPCC interface clock enable

Set and cleared by software.

0: IPCC clock disable for CPU1

1: IPCC clock enable for CPU1

- Bit 19 **HSEMEN** CPU1 HSEM clock enable  
Set and cleared by software.  
0: HSEM clock disable for CPU1  
1: HSEM clock enable for CPU1
- Bit 18 **RNGEN** CPU1 true RNG clocks enable  
Set and cleared by software.  
0: True RNG bus and kernel clocks disable for CPU1  
1: True RNG bus and kernel clocks enable for CPU1
- Bit 17 **AES2EN** CPU1 AES2 accelerator clock enable  
Set and cleared by software.  
0: AES2 clock disable for CPU1  
1: AES2 clock enable for CPU1
- Bit 16 **PKAEN** CPU1 PKA accelerator clock enable  
Set and cleared by software.  
0: PKA clock disable for CPU1  
1: PKA clock enable for CPU1
- Bits 15:9 Reserved, must be kept at reset value.
- Bit 8 **QUADSPIEN**: CPU1 Quad SPI memory interface clock enable  
Set and cleared by software.  
0: QUADSPI clock disable for CPU1  
1: QUADSPI clock enable for CPU1
- Bits 7:0 Reserved, must be kept at reset value.

#### 8.4.20 RCC APB1 peripheral clock enable register 1 (RCC\_APB1ENR1)

Address: 0x058

Reset value: 0x0000 0400

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

**Note:** *When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 EN	Res.	Res.	Res.	Res.	USB EN	Res.	CRS EN	I2C3 EN	Res.	I2C1 EN	Res.	Res.	Res.	Res.	Res.
rw					rw		rw	rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2 EN	Res.	Res.	WWWDG EN	RTCAPB EN	LCD EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2 EN
	rw			rs	rw	rw									rw

Bit 31 **LPTIM1EN**: CPU1 Low power timer 1 clocks enable

Set and cleared by software.

0: LPTIM1 bus and kernel clocks disabled for CPU1

1: LPTIM1 bus and kernel clocks enabled for CPU1

Bits 30:27 Reserved, must be kept at reset value.

- Bit 26 **USBEN**: CPU1 USB FS clocks enable  
Set and cleared by software  
0: USB FS bus and kernel clocks disabled for CPU1  
1: USB FS bus and kernel clocks enabled for CPU1
- Bits 25 Reserved, must be kept at reset value.
- Bit 24 **CRSEN**: CPU1 CRS clock enable  
Set and cleared by software.  
0: CRS clock disabled for CPU1  
1: CRS clock enabled for CPU1
- Bit 23 **I2C3EN**: CPU1 I2C3 clocks enable  
Set and cleared by software.  
0: I2C3 bus and kernel clocks disabled for CPU1  
1: I2C3 bus and kernel clocks enabled for CPU1
- Bits 22 Reserved, must be kept at reset value.
- Bit 21 **I2C1EN**: CPU1 I2C1 clocks enable  
Set and cleared by software.  
0: I2C1 bus and kernel clocks disabled for CPU1  
1: I2C1 bus and kernel clocks enabled for CPU1
- Bits 20:15 Reserved, must be kept at reset value.
- Bit 14 **SPI2EN**: CPU1 SPI2 clock enable (STM32WB55xx only)  
Set and cleared by software.  
0: SPI2 clock disabled for CPU1  
1: SPI2 clock enabled for CPU1  
Note that this bit is reserved on STM32WB35xx.
- Bits 13:12 Reserved, must be kept at reset value.
- Bit 11 **WWDGEN**: CPU1 Window watchdog clock enable  
Set by software to enable the window watchdog clock. Reset by hardware system reset.  
This bit can also be set by hardware if the WWDG\_SW option bit is reset.  
0: Window watchdog clock disabled for CPU1  
1: Window watchdog clock enabled for CPU1
- Bit 10 **RTCAPBEN**: CPU1 RTC APB clock enable  
Set and cleared by software  
0: RTC APB clock disabled for CPU1  
1: RTC APB clock enabled for CPU1
- Bit 9 **LCDEN**: CPU1 LCD clock enable (STM32WB55xx only)  
Set and cleared by software.  
0: LCD clock disabled for CPU1  
1: LCD clock enabled for CPU1  
Note that this bit is reserved on STM32WB35xx.
- Bits 8:1 Reserved, must be kept at reset value.
- Bit 0 **TIM2EN**: CPU1 TIM2 timer clock enable  
Set and cleared by software.  
0: TIM2 clock disabled for CPU1  
1: TIM2 clock enabled for CPU1

### 8.4.21 RCC APB1 peripheral clock enable register 2 (RCC\_APB1ENR2)

Address offset: 0x05C

Reset value: 0x00000 0000

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

**Note:** *When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPTIM2 EN	Res.	Res.	Res.	Res.	LPUART1 EN									
										rw					rw

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

Bit 5 **LPTIM2EN** CPU1 Low power timer 2 clocks enable

Set and cleared by software.

0: LPTIM2 bus and kernel clocks disabled for CPU1

1: LPTIM2 bus and kernel clocks enabled for CPU1

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **LPUART1EN**: CPU1 Low power UART 1 clocks enable

Set and cleared by software.

0: LPUART1 bus and kernel clocks disabled for CPU1

1: LPUART1 bus and kernel clocks enabled for CPU1

### 8.4.22 RCC APB2 peripheral clock enable register (RCC\_APB2ENR)

Address: 0x060

Reset value: 0x0000 0000

Access: word, half-word and byte access

**Note:** *When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1 EN	Res.	Res.	TIM17 EN	TIM16 EN	Res.
										rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 EN	Res.	SPI11 EN	TIM1 EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

Bits 31:22 Reserved, must be kept at reset value.

- Bit 21 **SAI1EN**: CPU1 SAI1 clocks enable  
Set and cleared by software.  
0: SAI1 bus and kernel clocks disabled for CPU1  
1: SAI1 bus and kernel clocks enabled for CPU1
- Bits 20:19 Reserved, must be kept at reset value.
- Bit 18 **TIM17EN**: CPU1 TIM17 timer clock enable  
Set and cleared by software.  
0: TIM17 timer clock disabled for CPU1  
1: TIM17 timer clock enabled for CPU1
- Bit 17 **TIM16EN**: CPU1 TIM16 timer clock enable  
Set and cleared by software.  
0: TIM16 timer clock disabled for CPU1  
1: TIM16 timer clock enabled for CPU1
- Bits 16:15 Reserved, must be kept at reset value.
- Bit 14 **USART1EN**: CPU1 USART1 clocks enable  
Set and cleared by software.  
0: USART1 bus and kernel clocks disabled for CPU1  
1: USART1 bus and kernel clocks enabled for CPU1
- Bit 13 Reserved, must be kept at reset value.
- Bit 12 **SPI1EN**: CPU1 SPI1 clock enable  
Set and cleared by software.  
0: SPI1 clock disabled for CPU1  
1: SPI1 clock enabled for CPU1
- Bit 11 **TIM1EN**: CPU1 TIM1 timer clock enable  
Set and cleared by software.  
0: TIM1 timer clock disabled for CPU1  
1: TIM1P timer clock enabled for CPU1
- Bits 10:0 Reserved, must be kept at reset value.

#### 8.4.23 RCC AHB1 peripheral clocks enable in Sleep modes register (RCC\_AHB1SMENR)

Address offset: 0x068

Reset value: 0x0001 1207

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC SMEN
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRC SMEN	Res.	Res.	SRAM1 SMEN	Res.	Res.	Res.	Res.	Res.	Res.	DMAMUX1 SMEN	DMA2 SMEN	DMA1 SMEN
			rw			rw							rw	rw	rw

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

- Bit 16 **TSCSMEN**: Touch sensing controller clock enable during CPU1 CSleep mode (STM32WB55xx only)  
 Set and cleared by software.  
 0: TSC clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: TSC clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode  
 Note that this bit is reserved on STM32WB35xx.

Bits 15:13 Reserved, must be kept at reset value.

- Bit 12 **CRCSMEN**: CRC clock enable during CPU1 CSleep mode  
 Set and cleared by software.  
 0: CRC clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: CRC clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bits 11:10 Reserved, must be kept at reset value.

- Bit 9 **SRAM1SMEN**: SRAM1 interface clock enable during CPU1 CSleep mode  
 Set and cleared by software.  
 0: SRAM1 interface clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: SRAM1 interface clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bits 8:3 Reserved, must be kept at reset value.

- Bit 2 **DMAMUX1SMEN**: DMAMUX1 clock enable during CPU1 CSleep mode  
 Set and cleared by software during Sleep mode.  
 0: DMAMUX1 clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: DMAMUX1 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

- Bit 1 **DMA2SMEN**: DMA2 clock enable during CPU1 CSleep mode  
 Set and cleared by software during Sleep mode.  
 0: DMA2 clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: DMA2 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

- Bit 0 **DMA1SMEN**: DMA1 clock enable during CPU1 CSleep mode  
 Set and cleared by software.  
 0: DMA1 clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: DMA1 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

#### 8.4.24 RCC AHB2 peripheral clocks enable in Sleep modes register (RCC\_AHB2SMENR)

Address offset: 0x06C

Reset value: 0x0001 209F

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AES1 SMEN
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADC SMEN	Res.	Res.	Res.	Res.	Res.	GPIOH SMEN	Res.	Res.	GPIOE SMEN	GPIOD SMEN	GPIOC SMEN	GPIOB SMEN	GPIOA SMEN
		rw						rw			rw	rw	rw	rw	rw

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

Bit 16 **AES1SMEN**: AES1 accelerator clock enable during CPU1 CSleep mode

Set and cleared by software.

0: AES1 clock disabled by the clock gating during CPU1 Csleep and CStop modes

1: AES1 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **ADCSMEN**: ADC clock enable during CPU1 Csleep and CStop modes

Set and cleared by software.

0: ADC bus clock disabled by the clock gating during CPU1 Csleep and CStop modes.

1: ADC bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHSMEN**: IO port H clock enable during CPU1 CSleep mode

Set and cleared by software.

0: IO port H clock disabled by the clock gating during CPU1 Csleep and CStop modes

1: IO port H clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **GPIOESMEN**: IO port E clock enable during CPU1 CSleep mode

Set and cleared by software.

0: IO port E clock disabled by the clock gating during CPU1 Csleep and CStop modes

1: IO port E clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bit 3 **GPIODSMEN**: IO port D clock enable during CPU1 CSleep mode (STM32WB55xx only)

Set and cleared by software.

0: IO port D clock disabled by the clock gating during CPU1 Csleep and CStop modes

1: IO port D clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Note that this bit is reserved on STM32WB35xx.

Bit 2 **GPIOCSMEN**: IO port C clock enable during CPU1 CSleep mode

Set and cleared by software.

0: IO port C clock disabled by the clock gating during CPU1 Csleep and CStop modes

1: IO port C clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

- Bit 1 **GPIOBSMEN**: IO port B clock enable during CPU1 CSleep mode  
 Set and cleared by software.  
 0: IO port B clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: IO port B clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode
- Bit 0 **GPIOASMEN**: IO port A clock enable during CPU1 CSleep mode  
 Set and cleared by software.  
 0: IO port A clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: IO port A clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

#### 8.4.25 RCC AHB3 and AHB4 peripheral clocks enable in Sleep and Stop modes register (RCC\_AHB3SMENR)

Address offset: 0x070

Reset value: 0x0307 0100

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLASH SMEN	SRAM2S MEN	Res.	Res.	Res.	Res.	Res.	RNG SMEN	AES2 SMEN	PKA SMEN
						rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QUADSPI SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
							rw								

Bits 31:26 Reserved, must be kept at reset value.

- Bit 25 **FLASHSMEN**: Flash memory interface clock enable during CPU1 CSleep mode  
 Set and cleared by software.  
 0: Flash interface clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: Flash interface clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

- Bit 24 **SRAM2SMEN**: SRAM2a and SRAM2b memory interface clock enable during CPU1 CSleep mode  
 Set and cleared by software.  
 0: SRAM2 clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: SRAM2 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

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

- Bit 18 **RNGSMEN**: True RNG clock enable during CPU1 Csleep and CStop modes  
 Set and cleared by software.  
 0: True RNG bus clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: True RNG bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

- Bit 17 **AES2MEN**: AES2 accelerator clock enable during CPU1 CSleep mode  
Set and cleared by software.  
0: AES2 clock disabled by the clock gating during CPU1 Csleep and CStop modes  
1: AES2 clock enabled by the clock gating during CPU1 CSleep mode, .disabled during CPU1 CStop mode
- Bit 16 **PKASMEN**: PKA accelerator clock enable during CPU1 CSleep mode  
Set and cleared by software.  
0: PKA clock disabled by the clock gating during CPU1 Csleep and CStop modes  
1: PKA clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.
- Bits 15:9 Reserved, must be kept at reset value.
- Bit 8 **QUADSPISMEN** Quad SPI memory interface clock enable during CPU1 CSleep mode  
Set and cleared by software.  
0: QUADSPI clock disabled by the clock gating during CPU1 Csleep and CStop modes  
1: QUADSPI clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.
- Bits 7:0 Reserved, must be kept at reset value.

#### 8.4.26 RCC APB1 peripheral clocks enable in Sleep mode register 1 (RCC\_APB1SMENR1)

Address: 0x078

Reset value: 0x85A0 4E01

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
LPTIM1 SMEN	Res.	Res.	Res.	Res.	USB SMEN	Res.	CRS SMEN	I2C3 SMEN	Res.	I2C1 SMEN	Res.	Res.	Res.	Res.	Res.
rw					rw		rw	rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2 SMEN	Res.	Res.	WWDG SMEN	RTCAPB SMEN	LCD SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2 SMEN
	rw			rw	rw	rw									rw

- Bit 31 **LPTIM1SMEN**: Low power timer 1 clock enable during CPU1 Csleep and CStop mode  
Set and cleared by software.  
0: LPTIM1 bus clock disabled by the clock gating during CPU1 Csleep and CStop modes  
1: LPTIM1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.
- Bits 30:27 Reserved, must be kept at reset value.
- Bit 26 **USBSMEN**: USB FS clock enable during CPU1 Csleep and CStop modes  
Set and cleared by software.  
0: USB FS bus clock disabled by the clock gating during CPU1 Csleep and CStop modes  
1: USB FS bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.
- Bits 25 Reserved, must be kept at reset value.

- Bit 24 **CRSSMEN:** CRS clock enable during CPU1 CSleep mode  
 Set and cleared by software.  
 0: CRS clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: CRS clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.
- Bit 23 **I2C3SMEN:** I2C3 clock enable during CPU1 Csleep and CStop modes  
 Set and cleared by software.  
 0: I2C3 bus clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: I2C3 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.
- Bits 22 Reserved, must be kept at reset value.
- Bit 21 **I2C1SMEN:** I2C1 clock enable during CPU1 Csleep and CStop modes  
 Set and cleared by software.  
 0: I2C1 bus clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: I2C1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.
- Bits 20:15 Reserved, must be kept at reset value.
- Bit 14 **SPI2SMEN:** SPI2 clock enable during CPU1 CSleep mode (STM32WB55xx only)  
 Set and cleared by software.  
 0: SPI2 clock disabled by the clock gating during CPU1 Csleep and CStop mode  
 1: SPI2 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.  
 Note that this bit is reserved on STM32WB35xx.
- Bits 13:12 Reserved, must be kept at reset value.
- Bit 11 **WWDGSMEN:** Window watchdog clocks enable during CPU1 CSleep mode  
 Set and cleared by software. This bit is forced to '1' by hardware when the hardware WWDG\_SW option is reset.  
 0: Window watchdog clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: Window watchdog clocks enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.
- Bit 10 **RTCAPBSMEN:** RTC bus clock enable during CPU1 CSleep mode  
 Set and cleared by software  
 0: RTC bus clock disabled during by the clock gating during CPU1 Csleep and CStop modes.  
 1: RTC bus clock enabled during by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.
- Bit 9 **LCDSMEN:** LCD clock enable during CPU1 CSleep mode (STM32WB55xx only)  
 Set and cleared by software.  
 0: LCD clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: LCD clocks enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.  
 Note that this bit is reserved on STM32WB35xx.
- Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **TIM2SMEN**: TIM2 timer clock enable during CPU1 CSleep mode  
 Set and cleared by software.  
 0: TIM2 clock disabled by the clock gating during CPU1 Csleep and CStop modes  
 1: TIM2 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

#### 8.4.27 RCC APB1 peripheral clocks enable in Sleep mode register 2 (RCC\_APB1SMENR2)

Address offset: 0x07C

Reset value: 0x0000 0021

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPTIM2 SMEN	Res.	Res.	Res.	Res.	LPUART1 SMEN									
										rw					rw

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

Bit 5 **LPTIM2SMEN** Low power timer 2 clock enable during CPU1 Csleep and CStop modes

Set and cleared by software.  
 0: LPTIM2 bus clock disabled by the clock gating during CPU1 Csleep and CStop modes.  
 1: LPTIM2 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **LPUART1SMEN**: Low power UART 1 clock enable during CPU1 Csleep and CStop modes.

Set and cleared by software.  
 0: LPUART1 bus clock disabled by the clock gating during CPU1 CSleep and CStop modes  
 1: LPUART1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

#### 8.4.28 RCC APB2 peripheral clocks enable in Sleep mode register (RCC\_APB2SMENR)

Address: 0x080

Reset value: 0x0026 5800

Access: word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1 SMEN	Res.	Res.	TIM17 SMEN	TIM16 SMEN	Res.
										rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 SMEN	Res.	SPI1 SMEN	TIM1 SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **SAI1SMEN**: SAI1 clock enable during CPU1 Csleep and CStop modes

Set and cleared by software.

0: SAI1 bus clock disabled by the clock gating during CPU1 Csleep and CStop mode

1: SAI1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17SMEN**: TIM17 timer clock enable during CPU1 CSleep mode

Set and cleared by software.

0: TIM17 timer clock disabled by the clock gating during CPU1 Csleep and CStop mode

1: TIM17 timer clocks enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 17 **TIM16SMEN**: TIM16 timer clock enable during CPU1 CSleep mode

Set and cleared by software.

0: TIM16 timer clock disabled by the clock gating during CPU1 Csleep and CStop mode

1: TIM16 timer clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1SMEN**: USART1 clock enable during CPU1 Csleep and CStop modes.

Set and cleared by software.

0: USART1 bus clock disabled by the clock gating during CPU1 Csleep and CStop mode

1: USART1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1SMEN**: SPI1 clock enable during CPU1 CSleep mode

Set and cleared by software.

0: SPI1 clock disabled by the clock gating during CPU1 Csleep and CStop mode

1: SPI1 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 11 **TIM1SMEN**: TIM1 timer clock enable during CPU1 CSleep mode

Set and cleared by software.

0: TIM1 timer clock disabled by the clock gating during CPU1 Csleep and CStop mode

1: TIM1 timer clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 10:0 Reserved, must be kept at reset value.

### 8.4.29 RCC peripherals independent clock configuration register (RCC\_CCIPR)

Address: 0x088

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
RNGSEL[1:0]	ADCSEL[1:0]	CLK48SEL[1:0]	Res.	Res.	SAI1SEL[1:0]	LPTIM2SEL[1:0]	LPTIM1SEL[1:0]	I2C3SEL[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	
Res.	Res.	I2C1SEL[1:0]	LPUART1SEL[1:0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1SEL[1:0]	
		rw	rw	rw											rw	rw

#### Bits 31:30 **RNGSEL[1:0]**: RNG clock source selection

These bits are set and cleared by software to select the clock source used by the true RNG.

00: Use clock as selected by CLK48SEL

01: Select LSI clock

10: Select LSE clock

11: Reserved

#### Bits 29:28 **ADCSEL[1:0]**: ADC clock source selection

These bits are set and cleared by software to select the clock source used by the ADC interface.

00: No clock selected

01: PLLSAI1 “R” clock (PLLSAI1RCLK) selected as ADC clock

10: PLL “P” clock (PLLPCLK) selected as ADC clock

11: System clock (SYSCLK) selected as ADC clock

#### Bits 27:26 **CLK48SEL[1:0]**: 48 MHz clock source selection

These bits are set and cleared by software to select the 48 MHz clock source used by USB and true RNG. The true RNG clock source selection is furthermore selected by RNGSEL.

00: HSI48 clock selected as 48 MHz clock

01: PLLSAI1 “Q” clock (PLLSAI1QCLK) selected as 48 MHz clock

10: PLL “Q” clock (PLLQCLK) selected as 48 MHz clock

11: MSI clock selected as 48 MHz clock

#### Bits 25:24 Reserved, must be kept at reset value.

#### Bits 23:22 **SAI1SEL[1:0]**: SAI1 clock source selection

These bits are set and cleared by software to select the SAI1 clock source.

00: PLLSAI1 “P” clock (PLLSAI1PCLK) selected as SAI1 clock

01: PLL “P” clock (PLLPCLK) selected as SAI1 clock

10: HSI16 clock selected as SAI1 clock

11: External input SAI1\_EXTCLK selected as SAI1 clock

**Caution:** If the selected clock is the external clock, it is not possible to switch to another clock if the external clock is not present.

Bits 21:20 **LPTIM2SEL[1:0]**: Low power timer 2 clock source selection

These bits are set and cleared by software to select the LPTIM2 clock source.

- 00: PCLK selected as LPTIM2 clock
- 01: LSI clock selected as LPTIM2 clock
- 10: HSI16 clock selected as LPTIM2 clock
- 11: LSE clock selected as LPTIM2 clock

Bits 19:18 **LPTIM1SEL[1:0]**: Low power timer 1 clock source selection

These bits are set and cleared by software to select the LPTIM1 clock source.

- 00: PCLK selected as LPTIM1 clock
- 01: LSI clock selected as LPTIM1 clock
- 10: HSI16 clock selected as LPTIM1 clock
- 11: LSE clock selected as LPTIM1 clock

Bits 17:16 **I2C3SEL[1:0]**: I2C3 clock source selection

These bits are set and cleared by software to select the I2C3 clock source.

- 00: PCLK selected as I2C3 clock
- 01: System clock (SYSCLK) selected as I2C3 clock
- 10: HSI16 clock selected as I2C3 clock
- 11: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **I2C1SEL[1:0]**: I2C1 clock source selection

These bits are set and cleared by software to select the I2C1 clock source.

- 00: PCLK selected as I2C1 clock
- 01: System clock (SYSCLK) selected as I2C1 clock
- 10: HSI16 clock selected as I2C1 clock
- 11: reserved

Bits 11:10 **LPUART1SEL[1:0]**: LPUART1 clock source selection

These bits are set and cleared by software to select the LPUART1 clock source.

- 00: PCLK selected as LPUART1 clock
- 01: System clock (SYSCLK) selected as LPUART1 clock
- 10: HSI16 clock selected as LPUART1 clock
- 11: LSE clock selected as LPUART1 clock

Bits 9:2 Reserved, must be kept at reset value.

Bits 1:0 **USART1SEL[1:0]**: USART1 clock source selection

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

- 00: PCLK selected as USART1 clock
- 01: System clock (SYSCLK) selected as USART1 clock
- 10: HSI16 clock selected as USART1 clock
- 11: LSE clock selected as USART1 clock

#### 8.4.30 RCC backup domain control register (RCC\_BDCR)

Address offset: 0x090

Reset value: 0x0000 0000, (reset by Backup domain reset, except LSCOSEL, LSCOEN and BDRST, which are reset only by Backup domain power-on reset, not reset by wakeup from Standby and System reset)

Access: 0 ≤ wait state ≤3, word, half-word and byte access.

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

**Note:** The bits of the [RCC backup domain control register \(RCC\\_BDCR\)](#) are outside of the  $V_{CORE}$  domain. As a result, after Reset, these bits are write-protected and the DBP bit in the [PWR control register 1 \(PWR\\_CR1\)](#) has to be set before these can be modified. Refer to [Section 6.1.4: Battery backup domain](#) for further information. These bits (except LSCOSEL, LSCOEN and BDRST) are only reset after a [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
Res.	Res.	Res.	Res.	Res.	Res.	LSCO SEL	LSCO EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST
						rw	rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		Res.	LSE CSSD	LSE CSSON	LSEDRV[1:0]		LSE BYP	LSE RDY	LSEON
rw						rw	rw		r	rw	rw	rw	rw	r	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **LSCOSEL:** Low speed clock output selection

Set and cleared by software.

- 0: LSI clock selected
- 1: LSE clock selected

Bit 24 **LSCOEN:** Low speed clock output enable

Set and cleared by software.

- 0: Low speed clock output (LSCO) disable
- 1: Low speed clock output (LSCO) enable

Bits 23:17 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

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, or unless a failure is detected on LSE (LSECSSD is set). The BDRST bit can be used to reset them.

- 00: No clock
- 01: LSE oscillator clock used as RTC clock
- 10: LSI oscillator clock used as RTC clock
- 11: HSE oscillator clock divided by 32 used as RTC clock

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LSECSSD** CSS on LSE failure detection

Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator (LSE).

- 0: No failure detected on LSE (32 kHz oscillator)
- 1: Failure detected on LSE (32 kHz oscillator)

Bit 5 **LSECSSON** CSS on LSE enable

Set by software to enable the clock security system on LSE (32 kHz oscillator).

LSECSSON must be enabled after the LSE oscillator is enabled (LSEON bit enabled) and ready (LSERDY flag set by hardware), and after the RTCSEL bit is selected.

Once enabled this bit cannot be disabled, except after a LSE failure detection (LSECSSD = 1). In that case the software MUST disable the LSECSSON bit.

- 0: CSS on LSE (32 kHz external oscillator) OFF
- 1: CSS on LSE (32 kHz external oscillator) ON

Bits 4:3 **LSEDRV[1:0]** LSE oscillator drive capability

Set by software to modulate the LSE oscillator drive capability.

- 00: 'Xtal mode' lower driving capability
- 01: 'Xtal mode' medium low driving capability
- 10: 'Xtal mode' medium high driving capability
- 11: 'Xtal mode' higher driving capability

The oscillator is in Xtal mode when it is not in bypass mode.

Bit 2 **LSEBYP**: LSE oscillator bypass

Set and cleared by software to bypass oscillator. This bit can be written only when the external 32 kHz oscillator is disabled (LSEON = 0 and LSERDY = 0).

- 0: LSE oscillator not bypassed
- 1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready

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

- 0: LSE oscillator not ready
- 1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enable

Set and cleared by software.

- 0: LSE oscillator OFF
- 1: LSE oscillator ON

### 8.4.31 RCC control/status register (RCC\_CSR)

Address: 0x094

Reset value: 0x0C00 0000 (reset by System reset, except reset flags by POR only, not reset by wakeup from Standby)

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWWDG RSTF	IWWG RSTF	SFT RSTF	BOR RSTF	PIN RSTF	OB L RSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	RF RSTS
r	r	r	r	r	r	r		rw							r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFWKSEL[1:0]		Res.	Res.	LSI2TRIM[3:0]				Res.	Res.	Res.	Res.	LSI2 RDY	LSI2 ON	LSI1 RDY	LSI1 ON
rw	rw			rw	rw	rw	rw					r	rw	r	rw

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

Set by hardware when a reset occurs due to illegal Stop, Standby or Shutdown mode entry.  
Cleared by writing to the RMVF bit.

0: No illegal mode reset occurred  
1: Illegal mode reset occurred

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

Set by hardware when a window watchdog reset occurs.  
Cleared by writing to the RMVF bit.  
0: No window watchdog reset occurred  
1: Window watchdog reset occurred

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

Set by hardware when an independent watchdog reset domain occurs.  
Cleared by writing to the RMVF bit.  
0: No independent watchdog reset occurred  
1: Independent watchdog reset occurred

Bit 28 **SFTRSTF**: Software reset flag

Set by hardware when a software reset occurs.  
Cleared by writing to the RMVF bit.  
0: No software reset occurred  
1: Software reset occurred

Bit 27 **BORRSTF**: BOR flag

Set by hardware when a BOR occurs.  
Cleared by writing to the RMVF bit.  
0: No BOR occurred  
1: BOR occurred

Bit 26 **PINRSTF**: Pin reset flag

Set by hardware when a reset from the NRST pin occurs.  
Cleared by writing to the RMVF bit.  
0: No reset from NRST pin occurred  
1: Reset from NRST pin occurred

Bit 25 **OBLRSTF**: Option byte loader reset flag

Set by hardware when a reset from the Option Byte loading occurs.  
Cleared by writing to the RMVF bit.  
0: No reset from Option Byte loading occurred  
1: Reset from Option Byte loading occurred

Bit 24 Reserved, must be kept at reset value.

Bit 23 **RMVF**: Remove reset flag

Set by software to clear the reset flags.

0: No effect

1: Clear the reset flags

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **RFRSTS**: Radio system BLE and 802.15.4 reset status

Set and cleared by hardware

0: Radio system BLE and 802.15.4 not in reset, radio system can be accessed

1: Radio system BLE and 802.15.4 under reset, radio system cannot be accessed

Bits 15:14 **RFWKPSEL[1:0]**: RF system wakeup clock source selection

Set by software to select the clock source for RF system wakeup logic.

00: No clock

01: LSE oscillator clock used as RF system wakeup clock

10: Reserved

11: HSE oscillator clock divided by 1024 used as RF system wakeup clock

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

Bits 11:8 **LSI2TRIM[3:0]**: LSI2 oscillator trim.

*Note: LSI2TRIM must be changed only when LSI2 is disabled (LSI2ON = 0).*

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

Bit 3 **LSI2RDY**: LSI2 oscillator ready

Set and cleared by hardware to indicate when the LSI2 oscillator is stable. After the LSI2ON bit is cleared, LSI2RDY goes low after three LSI2 oscillator clock cycles.

0: LSI2 oscillator not ready

1: LSI2 oscillator ready

Bit 2 **LSI2ON**: LSI2 oscillator enable and selection

Set and cleared by software.

0: LSI2 oscillator OFF (LSI1 selected on LSI)

1: LSI2 oscillator ON(LSI2 when ready selected on LSI)

Bit 1 **LSI1RDY**: LSI1 oscillator ready

Set and cleared by hardware to indicate when the LSI1 oscillator is stable. After the LSI1ON bit is cleared, LSI1RDY goes low after three LSI1 oscillator clock cycles. This bit can be set even if LSI1ON = 0 if the LSI1 is requested by the clock security system on LSE, by the Independent watchdog or by the RTC.

0: LSI1 oscillator not ready

1: LSI1 oscillator ready

Bit 0 **LSI1ON**: LSI1 oscillator enable

Set and cleared by software.

0: LSI1 oscillator OFF

1: LSI1 oscillator ON

#### 8.4.32 RCC clock recovery RC register (RCC\_CRRCR)

Address: 0x098

Reset value: 0x0000 XXX0 where X is factory-programmed.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSI48CAL[8:0]										Res.	Res.	Res.	Res.	Res.	HSI48 RDY
r	r	r	r	r	r	r	r	r						r	rw

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

Bits 15:7 **HSI48CAL[8:0]**: HSI48 clock calibration

These bits are initialized at startup with the factory-programmed HSI48 calibration trim value. They are ready only.

Bits 6:2 Reserved, must be kept at reset value

Bit 1 **HSI48RDY**: HSI48 clock ready flag

Set by hardware to indicate that HSI48 oscillator is stable. This bit is set only when HSI48 is enabled by software by setting HSI48ON.

0: HSI48 oscillator not ready  
1: HSI48 oscillator ready

Bit 0 **HSI48ON**: HSI48 clock enable

Set and cleared by software.

Cleared by hardware to stop the HSI48 when entering in Stop, Standby or Shutdown modes.

0: HSI48 oscillator OFF  
1: HSI48 oscillator ON

#### 8.4.33 RCC clock HSE register (RCC\_HSECR)

Address: 0x09C

Reset value: 0x0000 0030 (reset by System reset, not reset by wakeup from Standby)

Access: Write access is protected and software must write a KEY (0xCAFE CAFE) as a word access in this register to unlock the register. Once unlocked a single write access (word, half-word or byte) can be performed to the register. It is then locked again. No wait states.

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

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

Bits 13:8 **HSETUNE[5:0]**: HSE capacitor tuning

Can be changed by software. Not to be changed when HSE is on.

0x00: minimum load capacitance  
0x3F maximum load capacitance

Bits 7 Reserved, must be kept at reset value

Bits 6:4 **HSEGMC[2:0]**: HSE current control

Can be changed by software. Not to be changed when HSE is on.

- 000: current max limit 0.18 mA/V
- 001: current max limit 0.57 mA/V
- 010: current max limit 0.78 mA/V
- 011: current max limit 1.13 mA/V
- 100: current max limit 0.61 mA/V
- 101: current max limit 1.65 mA/V
- 110: current max limit 2.12 mA/V
- 111: current max limit 2.84 mA/V

*Note: The HSEGMC value must be greater than  $g_{mcrit}$ , whose value must be calculated according to AN2867 "Oscillator design guide for STM8S, STM8A and STM32 microcontrollers", available on [www.st.com](http://www.st.com).*

Bit 3 **HSES**: HSE Sense amplifier threshold

Can be changed by software. Not to be changed when HSE is on.

- 0: HSE bias current factor 1/2
- 1: HSE bias current factor 3/4

## Bits 2:1 Reserved, must be kept at reset value

Bit 0 **UNLOCKED**: HSE clock control register unlocked

Set and cleared by hardware.

- 0: register locked, the key has not been programmed, or a write access has been performed to the register
- 1: Register unlocked. Key 0xCAFE CAFE has been written to unlock this register, enabling a single data write

**8.4.34 RCC extended clock recovery register (RCC\_EXTCFGR)**

Address: 0x108

Reset value: 0x0003 0000.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	RFCSS	Res.	Res.	C2HPREF	SHDHPREF								
											r			r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	C2HPRE[3:0]				SHDHPRE[3:0]										
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 20 **RFCSS**: Radio system HCLK5 and APB3 selected clock source indication

Set and reset by hardware to indicate which clock source is selected for the Radio system HCLK5 and APB3 clock.

- 0: HSI16 used for Radio system HCLK5 and APB3 clock
- 1: HSE divided by 2 used for Radio system HCLK5 and APB3 clock

## Bits 19:18 Reserved, must be kept at reset value

Bit 17 **C2HPREF**: HCLK2 prescaler flag (CPU2)

Set and reset by hardware to acknowledge HCLK2 prescaler programming

Reset when a new prescaler value is programmed in C2HPRE. set when the programmed value is actually applied.

0: HCLK2 prescaler value not yet applied

1: HCLK2 prescaler value applied

Bit 16 **SHDHPREF**: HCLK4 shared prescaler flag (AHB4, Flash memory and SRAM2)

Set and reset by hardware to acknowledge Shared HCLK4 prescaler programming

Reset when a new prescaler value is programmed in SHDHPRE. set when the programmed value is actually applied.

0: HCLK4 prescaler value not yet applied

1: HCLK4 prescaler value applied

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

Bits 7:4 **C2HPRE[3:0]**: HCLK2 prescaler (CPU2)

Set and cleared by software to control the division factor of the HCLK2 clock (CPU2).

The C2HPREF flag can be checked to know if the programmed C2HPRE prescaler value is applied.

**Caution:** Depending on the device voltage range, the software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details refer to [Section 6.1.6: Dynamic voltage scaling management](#)). After a write operation to these bits and before decreasing the voltage range the register bit C2HPREF must be read to be sure that the new value has been taken into account.

0001: SYSCLK divided by 3

0010: SYSCLK divided by 5

0101: SYSCLK divided by 6

0110: SYSCLK divided by 10

0111: SYSCLK divided by 32

1000: SYSCLK divided by 2

1001: SYSCLK divided by 4

1010: SYSCLK divided by 8

1011: SYSCLK divided by 16

1100: SYSCLK divided by 64

1101: SYSCLK divided by 128

1110: SYSCLK divided by 256

1111: SYSCLK divided by 512

Others: SYSCLK not divided

Bit 3:0 **SHDHPRE[3:0]**: HCLK4 shared prescaler (AHB4, Flash memory and SRAM2)

Set and cleared by software to control the division factor of the Shared HCLK4 clock (AHB4, Flash memory and SRAM2).

The SHDHPREF flag can be checked to know if the programmed SHDHPRE prescaler value is applied.

**Caution:** Depending on the device voltage range, the software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details refer to [Section 6.1.6: Dynamic voltage scaling management](#)). After a write operation to these bits and before decreasing the voltage range the register bit SHDHPRE must be read to be sure that the new value has been taken into account.

0001: SYSCLK divided by 3  
 0010: SYSCLK divided by 5  
 0101: SYSCLK divided by 6  
 0110: SYSCLK divided by 10  
 0111: SYSCLK divided by 32  
 1000: SYSCLK divided by 2  
 1001: SYSCLK divided by 4  
 1010: SYSCLK divided by 8  
 1011: SYSCLK divided by 16  
 1100: SYSCLK divided by 64  
 1101: SYSCLK divided by 128  
 1110: SYSCLK divided by 256  
 1111: SYSCLK divided by 512  
 Others: SYSCLK not divided

#### 8.4.35 RCC CPU2 AHB1 peripheral clock enable register (RCC\_C2AHB1ENR)

Address offset: 0x148

Reset value: 0x0000 0000

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

**Note:** *When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not supported.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCEN
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCEN	Res.	Res.	SRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	DMAMUX1EN	DMA2EN	DMA1EN
			rw			rw							rw	rw	rw

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

Bit 16 **TSCEN**: CPU2 Touch sensing controller clock enable (STM32WB55xx only)

Set and cleared by software.

0: TSC clock disable for CPU2

1: TSC clock enable for CPU2

Note that this bit is reserved on STM32WB35xx.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CPU2 CRC clock enable

Set and cleared by software.

0: CRC clock disable for CPU2

1: CRC clock enable for CPU2

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **SRAM1EN**: CPU2 SRAM1 clock enable

Set and cleared by software.

0: SRAM1 clock disabled for CPU2

1: SRAM1 clock enabled for CPU2

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1**: CPU2 DMAMUX1 clock enable

Set and cleared by software.

0: DMAMUX1 clock disable for CPU2

1: DMAMUX1 clock enable for CPU2

Bit 1 **DMA2EN**: CPU2 DMA2 clock enable

Set and cleared by software.

0: DMA2 clock disable for CPU2

1: DMA2 clock enable for CPU2

Bit 0 **DMA1EN**: CPU2 DMA1 clock enable

Set and cleared by software.

0: DMA1 clock disable for CPU2

1: DMA1 clock enable for CPU2

#### 8.4.36 RCC CPU2 AHB2 peripheral clock enable register (RCC\_C2AHB2ENR)

Address offset: 0x14C

Reset value: 0x0000 0000

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

**Note:** When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AES1 EN
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADC EN	Res.	Res.	Res.	Res.	Res.	GPIOH EN	Res.	Res.	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
		rw						rw			rw	rw	rw	rw	rw

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

- Bit 16 **AES1EN**: CPU2 AES1 accelerator clock enable  
 Set and cleared by software.  
 0: AES clock disabled for CPU2  
 1: AES clock enabled for CPU2
- Bits 15:14 Reserved, must be kept at reset value.
- Bit 13 **ADCEN**: ADC clocks enable  
 Set and cleared by software.  
 0: ADC bus and kernel clocks disabled for CPU2  
 1: ADC bus and kernel clocks enabled for CPU2
- Bits 12:8 Reserved, must be kept at reset value.
- Bit 7 **GPIOHEN**: CPU2 IO port H clock enable  
 Set and cleared by software.  
 0: IO port H clock disabled for CPU2  
 1: IO port H clock enabled for CPU2
- Bits 6:5 Reserved, must be kept at reset value.
- Bit 4 **GPIOEEN**: CPU2 IO port E clock enable  
 Set and cleared by software.  
 0: IO port E clock disabled for CPU2  
 1: IO port E clock enabled for CPU2
- Bit 3 **GPIODEN**: CPU2 IO port D clock enable (STM32WB55xx only)  
 Set and cleared by software.  
 0: IO port D clock disabled for CPU2  
 1: IO port D clock enabled for CPU2  
 Note that this bit is reserved on STM32WB35xx.
- Bit 2 **GPIOCEN**: CPU2 IO port C clock enable  
 Set and cleared by software.  
 0: IO port C clock disabled for CPU2  
 1: IO port C clock enabled for CPU2
- Bit 1 **GPIOBEN**: CPU2 IO port B clock enable  
 Set and cleared by software.  
 0: IO port B clock disabled for CPU2  
 1: IO port B clock enabled for CPU2
- Bit 0 **GPIOAEN**: CPU2 IO port A clock enable  
 Set and cleared by software.  
 0: IO port A clock disabled for CPU2  
 1: IO port A clock enabled for CPU2

#### 8.4.37 RCC CPU2 AHB3 and AHB4 peripheral clock enable register (RCC\_C2AHB3ENR)

Address offset: 0x150

Reset value: 0x0208 0000

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

*Note:* When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLASH EN	Res.	Res.	Res.	Res.	IPCC EN	HSEM EN	RNG EN	AES2 EN	PKA EN
						rw					rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						

Bits 31:26 Reserved, must be kept at reset value.

**Bit 25 FLASHEN** CPU2 Flash interface clock enable

This bit can only be cleared when the Flash is in Power down. Set and cleared by software.

- 0: Flash interface clock disable for CPU2
- 1: Flash interface clock enable for CPU2

Bits 24:21 Reserved, must be kept at reset value.

**Bit 20 IPCCEN** CPU2 IPCC interface clock enable

Set and cleared by software.

- 0: IPCC clock disable for CPU2
- 1: IPCC clock enable for CPU2

**Bit 19 HSEMEN** CPU2 HSEM clock enable

Set and cleared by software.

- 0: HSEM clock disable for CPU2
- 1: HSEM clock enable for CPU2

**Bit 18 RNGEN** CPU2 true RNG clocks enable

Set and cleared by software.

- 0: True RNG bus and kernel clocks disable for CPU2
- 1: True RNG bus and kernel clocks enable for CPU2

**Bit 17 AES2EN** CPU2 AES2 accelerator clock enable

Set and cleared by software.

- 0: AES2 clock disable for CPU2
- 1: AES2 clock enable for CPU2

**Bit 16 PKAEN** CPU2 PKA accelerator clock enable

Set and cleared by software.

- 0: PKA clock disable for CPU2
- 1: PKA clock enable for CPU2

Bits 15:0 Reserved, must be kept at reset value.

#### 8.4.38 RCC CPU2 APB1 peripheral clock enable register 1 (RCC\_C2APB1ENR1)

Address: 0x158

Reset value: 0x0000 0400

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

**Note:** *When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not supported.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 EN	Res.	Res.	Res.	Res.	USB EN	Res.	CRS EN	I2C3 EN	Res.	I2C1 EN	Res.	Res.	Res.	Res.	Res.
rw					rw		rw	rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2 EN	Res.	Res.	Res.	RTCAPB EN	LCD EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2 EN
	rw				rw	rw									rw

Bit 31 **LPTIM1EN**: CPU2 Low power timer 1 clocks enable

Set and cleared by software.

0: LPTIM1 bus and kernel clocks disabled for CPU2

1: LPTIM1 bus and kernel clocks enabled for CPU2

Bits 30:27 Reserved, must be kept at reset value.

Bit 26 **USBEN**: CPU2 USB FS clocks enable

Set and cleared by software

0: USB FS bus and kernel clocks disabled for CPU2

1: USB FS bus and kernel clocks enabled for CPU2

Bits 25 Reserved, must be kept at reset value.

Bit 24 **CRSEN**: CPU2 CRS clock enable

Set and cleared by software.

0: CRS clock disabled for CPU2

1: CRS clock enabled for CPU2

Bit 23 **I2C3EN**: CPU2 I2C3 clocks enable

Set and cleared by software.

0: I2C3 bus and kernel clocks disabled for CPU2

1: I2C3 bus and kernel clocks enabled for CPU2

Bits 22 Reserved, must be kept at reset value.

Bit 21 **I2C1EN**: CPU2 I2C1 clocks enable

Set and cleared by software.

0: I2C1 bus and kernel clocks disabled for CPU2

1: I2C1 bus and kernel clocks enabled for CPU2

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **SPI2EN**: CPU2 SPI2 clock enable (STM32WB55xx only)

Set and cleared by software.

0: SPI2 clock disabled for CPU2

1: SPI2 clock enabled for CPU2

Note that this bit is reserved on STM32WB35xx.

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

Bit 10 **RTCAPBEN**: CPU2 RTC APB clock enable

Set and cleared by software

0: RTC APB clock disabled for CPU2

1: RTC APB clock enabled for CPU2

Bit 9 **LCDEN**: CPU2 LCD clock enable (STM32WB55xx only)  
 Set and cleared by software.  
 0: LCD clock disabled for CPU2  
 1: LCD clock enabled for CPU2  
 Note that this bit is reserved on STM32WB35xx.

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **TIM2EN**: CPU2 TIM2 timer clock enable  
 Set and cleared by software.  
 0: TIM2 clock disabled for CPU2  
 1: TIM2 clock enabled for CPU2

#### 8.4.39 RCC CPU2 APB1 peripheral clock enable register 2 (RCC\_C2APB1ENR2)

Address offset: 0x15C

Reset value: 0x00000 0000

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

*Note:* When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPTIM2EN	Res.	Res.	Res.	Res.	LPUART1EN									
										rw					rw

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

Bit 5 **LPTIM2EN**: CPU2 Low power timer 2 clocks enable  
 Set and cleared by software.  
 0: LPTIM2 bus and kernel clocks disable for CPU2  
 1: LPTIM2 bus and kernel clocks enable for CPU2

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **LPUART1EN**: CPU2 Low power UART 1 clocks enable  
 Set and cleared by software.  
 0: LPUART1 bus and kernel clocks disable for CPU2  
 1: LPUART1 bus and kernel clocks enable for CPU2

#### 8.4.40 RCC CPU2 APB2 peripheral clock enable register (RCC\_C2APB2ENR)

Address: 0x160

Reset value: 0x0000 0000

Access: word, half-word and byte access

**Note:** When the peripheral clock is not active the peripheral registers read or write access from CPU2 is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1 EN	Res.	Res.	TIM17 EN	TIM16 EN	Res.
										rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 EN	Res.	SPI1 EN	TIM1 EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **SAI1EN:** CPU2 SAI1 clocks enable

Set and cleared by software.

0: SAI1 bus and kernel clocks disabled for CPU2

1: SAI1 bus and kernel clocks enabled for CPU2.

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN:** CPU2 TIM17 timer clock enable

Set and cleared by software.

0: TIM17 timer clock disabled for CPU2

1: TIM17 timer clock enabled for CPU2

Bit 17 **TIM16EN:** CPU2 TIM16 timer clock enable

Set and cleared by software.

0: TIM16 timer clock disabled for CPU2

1: TIM16 timer clock enabled for CPU2

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1EN:** CPU2 UFSART1 clock enable

Set and cleared by software.

0: USART1 bus and kernel clocks disabled for CPU2

1: USART1 bus and kernel clocks enabled for CPU2

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN:** CPU2 SPI1 clock enable

Set and cleared by software.

0: SPI1 clock disabled for CPU2

1: SPI1 clock enabled for CPU2

Bit 11 **TIM1EN:** CPU2 TIM1 timer clock enable

Set and cleared by software.

0: TIM1 timer clock disabled for CPU2

1: TIM1P timer clock enabled for CPU2

Bits 10:0 Reserved, must be kept at reset value.

#### 8.4.41 RCC CPU2 APB3 peripheral clock enable register (RCC\_C2APB3ENR)

Address offset: 0x164

Reset value: 0x00000 0000

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

**Note:** *When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not supported.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	802EN	BLEEN													
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **802EN:** CPU2 802.15.4 interface clock enable

Set and cleared by software.

0: 802.15.4 clock disable for CPU2

1: 802.15.4 clock enable for CPU2

Bit 0 **BLEEN:** CPU2 BLE interface clock enable

Set and cleared by software.

0: BLE clock disable for CPU2

1: BLE clock enable for CPU2

#### 8.4.42 RCC CPU2 AHB1 peripheral clocks enable in Sleep modes register (RCC\_C2AHB1SMENR)

Address offset: 0x168

Reset value: 0x0001 1007

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC SMEN	
															rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	Res.	Res.	CRC SMEN	Res.	Res.	SRAM1 SMEN	Res.	DMAMUX1 SMEN	DMA2 SMEN	DMA1 SMEN						
			rw			rw								rw	rw	rw

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

Bit 16 **TSCSMEN:** Touch sensing controller clock enable during CPU2 CSleep mode (STM32WB55xx only)

Set and cleared by software.

0: TSC clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: TSC clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Note that this bit is reserved on STM32WB35xx.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCSMEN**: CRC clock enable during CPU2 CSleep mode

Set and cleared by software.

0: CRC clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: CRC clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **SRAM1SMEN**: SRAM1 interface clock enabled during CPU2 CSleep mode

Set and cleared by software

0: SRAM1 interface clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: SRAM1 interface clock enabled by the clock gating during CPU2 CSleep mode, disabled during CStop mode

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUX1SMEN**: DMAMUX1 clock enable during CPU2 CSleep mode

Set and cleared by software during Sleep mode.

0: DMAMUX1 clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: DMAMUX1 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 1 **DMA2SMEN**: DMA2 clock enable during CPU2 CSleep mode

Set and cleared by software during Sleep mode.

0: DMA2 clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: DMA2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bit 0 **DMA1SMEN**: DMA1 clock enable during CPU2 CSleep mode

Set and cleared by software.

0: DMA1 clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: DMA1 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

#### 8.4.43 RCC CPU2 AHB2 peripheral clocks enable in Sleep modes register (RCC\_C2AHB2SMENR)

Address offset: 0x16C

Reset value: 0x0001 209F

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	AES1 SMEN
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	Res.	ADC SMEN	Res.	Res.	Res.	Res.	Res.	GPIOH SMEN	Res.	Res.	GPIOE SMEN	GPIOD SMEN	GPIOC SMEN	GPIOB SMEN	GPIOA SMEN	
		rw						rw			rw	rw	rw	rw	rw	

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

- Bit 16 **AES1SMEN**: AES1 accelerator clock enable during CPU2 CSleep mode  
 Set and cleared by software.  
 0: AES1 clock disabled by the clock gating during CPU2 CSleep and CStop modes.  
 1: AES1 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.
- Bits 15:14 Reserved, must be kept at reset value.
- Bit 13 **ADCSMEN**: ADC clock enable during CPU2 CSleep and CStop modes  
 Set and cleared by software.  
 0: ADC bus clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: ADC bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode
- Bits 12:8 Reserved, must be kept at reset value.
- Bit 7 **GPIOHSMEN**: IO port H clock enable during CPU2 CSleep mode  
 Set and cleared by software.  
 0: IO port H clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: IO port H clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode
- Bits 6:5 Reserved, must be kept at reset value.
- Bit 4 **GPIOESMEN**: IO port E clock enable during CPU2 CSleep mode  
 Set and cleared by software.  
 0: IO port E clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: IO port E clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode
- Bit 3 **GPIODSMEN**: IO port D clock enable during CPU2 CSleep mode (STM32WB55xx only)  
 Set and cleared by software.  
 0: IO port D clock disabled by the clock gating during CPU2 CSleep and CStop modes.  
 1: IO port D clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.  
 Note that this bit is reserved on STM32WB35xx.
- Bit 2 **GPIOCSMEN**: IO port C clock enable during CPU2 CSleep mode  
 Set and cleared by software.  
 0: IO port C clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: IO port C clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode
- Bit 1 **GPIOBSMEN**: IO port B clock enable during CPU2 CSleep mode  
 Set and cleared by software.  
 0: IO port B clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: IO port B clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode
- Bit 0 **GPIOASMEN**: IO port A clock enable during CPU2 CSleep mode  
 Set and cleared by software.  
 0: IO port A clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: IO port A clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

#### 8.4.44 RCC CPU2 AHB3 and AHB4 peripheral clocks enable in Sleep mode register (RCC\_C2AHB3SMENR)

Address offset: 0x170

Reset value: 0x0307 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLASH SMEN	SRAM2 SMEN	Res.	Res.	Res.	Res.	Res.	RNG SMEN	AES2 SMEN	PKA SMEN
						rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHSMEN** Flash memory interface clock enable during CPU2 CSleep mode

Set and cleared by software.

0: Flash memory interface clock disabled by the clock gating during CPU2 CSleep and CStop modes.

1: Flash memory interface clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bit 24 **SRAM2SMEN**: SRAM2a and SRAM2b interface clock enable during CPU2 CSleep mode

Set and cleared by software.

0: SRAM2 clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: SRAM2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

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

Bit 18 **RNGSMEN** True RNG clock enable during CPU2 CSleep and CStop mode

Set and cleared by software.

0: True RNG bus clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: True RNG bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 17 **AES2SMEN** AES2 accelerator clock enable during CPU2 CSleep mode

Set and cleared by software.

0: AES2 clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: AES2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 16 **PKASMEN**: PKA accelerator clock enable during CPU2 CSleep mode

Set and cleared by software.

0: PKA clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: PKA clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 15:0 Reserved, must be kept at reset value.

### 8.4.45 RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 1 (RCC\_C2APB1SMENR1)

Address: 0x178

Reset value: 0x85A0 4601

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
LPTIM1 SMEN	Res.	Res.	Res.	Res.	USB SMEN	Res.	CRS SMEN	I2C3 SMEN	Res.	I2C1 SMEN	Res.	Res.	Res.	Res.	Res.
rw					rw		rw	rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2 SMEN	Res.	Res.	Res.	RTCAPB SMEN	.LCD SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2 SMEN
	rw				rw	rw									rw

Bit 31 **LPTIM1SMEN**: Low power timer 1 clock enable during CPU2 CSleep and CStop mode

Set and cleared by software.

0: LPTIM1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: LPTIM1 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 30:27 Reserved, must be kept at reset value.

Bit 26 **USBSMEN**: USB FS clock enable during CPU2 CSleep and CStop modes

Set and cleared by software.

0: USB FS bus clock disabled by the clock gating during CPU2 CSleep and CStop modes.

1: USB FS bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bits 25 Reserved, must be kept at reset value.

Bit 24 **CRSSMEN**: CRS clock enable during CPU2 CSleep mode

Set and cleared by software.

0: CRS clock disabled by the clock gating during CPU2 CSleep and CStop modes.

1: CRS clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bit 23 **I2C3SMEN**: I2C3 clock enable during CPU2 CSleep and CStop modes

Set and cleared by software.

0: I2C3 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes.

1: I2C3 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bits 22 Reserved, must be kept at reset value.

Bit 21 **I2C1SMEN**: I2C1 clock enable during CPU2 CSleep and CStop modes

Set and cleared by software.

0: I2C1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: I2C1 bus clock enabled by the clock gating during CPU2 CSleep modes disabled during CPU2 CStop mode

Bits 20:15 Reserved, must be kept at reset value.

- Bit 14 **SPI2SMEN:** SPI2 clock enable during CPU2 CSleep mode (STM32WB55xx only)  
 Set and cleared by software.  
 0: SPI2 clock disabled by the clock gating during CPU2 CSleep and CStop modes.  
 1: SPI2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.  
 Note that this bit is reserved on STM32WB35xx.
- Bits 13:11 Reserved, must be kept at reset value.
- Bit 10 **RTCAPBSMEN:** RTC bus clock enable during CPU2 CSleep mode  
 Set and cleared by software  
 0: RTC bus clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: RTC bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode
- Bit 9 **LCDSMEN:** LCD clock enable during CPU2 CSleep mode (STM32WB55xx only)  
 Set and cleared by software.  
 0: LCD clock disabled by the clock gating during CPU2 CSleep and CStop modes.  
 1: LCD clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.  
 Note that this bit is reserved on STM32WB35xx.
- Bits 8:1 Reserved, must be kept at reset value.
- Bit 0 **TIM2SMEN:** TIM2 timer clock enable during CPU2 CSleep mode  
 Set and cleared by software.  
 0: TIM2 clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: TIM2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

#### 8.4.46 RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 2 (RCC\_C2APB1SMENR2)

Address offset: 0x17C

Reset value: 0x0000 0021

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPTIM2 SMEN	Res.	Res.	Res.	Res.	LPUART1 SMEN									
										rw					rw

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

Bit 5 **LPTIM2SMEN** Low power timer 2 clock enable during CPU2 CSleep and CStop modes.

Set and cleared by software.

0: LPTIM2 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes.

1: LPTIM2 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **LPUART1SMEN**: Low power UART 1 clock enable during CPU2 CSleep and CStop mode

Set and cleared by software.

0: LPUART1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes.

1: LPUART1 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

#### 8.4.47 RCC CPU2 APB2 peripheral clocks enable in Sleep mode register (RCC\_C2APB2SMENR)

Address: 0x180

Reset value: 0x0026 5800

Access: word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1 SMEN	Res.	Res.	TIM17 SMEN	TIM16 SMEN	Res.
										rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 SMEN	Res.	SPI1 SMEN	TIM1SM EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **SAI1SMEN**: SAI1 clock enable during CPU2 CSleep and CStop mode

Set and cleared by software.

0: SAI1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes.

1: SAI1 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17SMEN**: TIM17 timer clock enable during CPU2 CSleep mode

Set and cleared by software.

0: TIM17 timer clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: TIM17 timer clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 17 **TIM16SMEN**: TIM16 timer clock enable during CPU2 CSleep mode

Set and cleared by software.

0: TIM16 timer clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: TIM16 timer clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1SMEN**: USART1 clock enable during CPU2 CSleep and CStop mode

Set and cleared by software.

0: USART1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes

1: USART1 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 13 Reserved, must be kept at reset value.

- Bit 12 **SPI1SMEN**: SPI1 clock enable during CPU2 CSleep mode  
 Set and cleared by software.  
 0: SPI1 clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: SPI1 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode
- Bit 11 **TIM1SMEN**: TIM1 timer clock enable during CPU2 CSleep mode  
 Set and cleared by software.  
 0: TIM1 timer clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: TIM1 timer clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode
- Bits 10:0 Reserved, must be kept at reset value.

#### 8.4.48 RCC CPU2 APB3 peripheral clock enable in Sleep mode register (RCC\_C2APB3SMENR)

Address offset: 0x184

Reset value: 0x00000 0003

Access: word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	802SMEN	BLESMEN													
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

- Bit 1 **802SMEN**: 802.15.4 interface bus clock enable during CPU2 CSleep mode  
 Set and cleared by software.  
 0: 802.15.4 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: 802.15.4 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode
- Bit 0 **BLESMEN**: BLE interface bus clock enable during CPU2 CSleep mode  
 Set and cleared by software.  
 0: BLE bus clock disabled by the clock gating during CPU2 CSleep and CStop modes  
 1: BLE bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

*Note: The BLE interface bus clock is also enabled by the BLE IP itself.*

#### 8.4.49 RCC register map

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

**Table 42. RCC register map and reset values**

**Table 42. RCC register map and reset values (continued)**

**Table 42. RCC register map and reset values (continued)**

Offset	Register	Field	Type	Reset	Description
0x054	Reserved		Res.	0	LPTIM1EN
0x058	RCC_APB1ENR1		Res.	0	LPTIM1EN
0x05C	RCC_APB1ENR2		Res.	0	Res.
0x060	RCC_APB2ENR		Res.	0	Res.
0x064	Reserved		Res.	0	CRSEN
0x068	RCC_AHB1SMENR		Res.	0	I2C3EN
0x06C	RCC_AHB2SMENR		Res.	0	I2C1EN
0x070	RCC_AHB3SMENR		Res.	0	Res.
0x074	Reserved		Res.	0	Res.
0x078	RCC_APB1SMENR1		Res.	0	Res.
0x07C	RCC_APB1SMENR2		Res.	0	Res.
0x080	RCC_APB2SMENR		Res.	0	Res.
0x084	RCC_APB3SMENR		Res.	0	Res.
0x088	RCC_DCDEN		Res.	0	Res.
0x08C	RCC_FMCEN		Res.	0	Res.
0x090	RCC_HSEBYP		Res.	0	Res.
0x094	RCC_HSEEN		Res.	0	Res.
0x098	RCC_HSIEN		Res.	0	Res.
0x09C	RCC_I2C1EN		Res.	0	Res.
0x0A0	RCC_I2C2EN		Res.	0	Res.
0x0A4	RCC_I2C3EN		Res.	0	Res.
0x0A8	RCC_LPTIM1EN		Res.	0	Res.
0x0AC	RCC_LPTIM2EN		Res.	0	Res.
0x0B0	RCC_LTDCEN		Res.	0	Res.
0x0B4	RCC_MAMUX1SMEN		Res.	0	Res.
0x0B8	RCC_DMA2SMEN		Res.	0	Res.
0x0BC	RCC_GPIOASMEN		Res.	0	Res.
0x0C0	RCC_DMA1SMEN		Res.	0	Res.
0x0C4	RCC_LPUART1EN		Res.	0	Res.
0x0C8	RCC_TIM2EN		Res.	0	Res.

**Table 42. RCC register map and reset values (continued)**

**Table 42. RCC register map and reset values (continued)**

**Table 42. RCC register map and reset values (continued)**

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

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

### 9.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIO<sub>x</sub>\_MODER, GPIO<sub>x</sub>\_OTYPER, GPIO<sub>x</sub>\_OSPEEDR and GPIO<sub>x</sub>\_PUPDR), two 32-bit data registers (GPIO<sub>x</sub>\_IDR and GPIO<sub>x</sub>\_ODR) and a 32-bit set/reset register (GPIO<sub>x</sub>\_BSRR). In addition all GPIOs have a 32-bit locking register (GPIO<sub>x</sub>\_LCKR) and two 32-bit alternate function selection registers (GPIO<sub>x</sub>\_AFRH and GPIO<sub>x</sub>\_AFRL).

### 9.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIO<sub>x</sub>\_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIO<sub>x</sub>\_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIO<sub>x</sub>\_BSRR) for bitwise write access to GPIO<sub>x</sub>\_ODR
- Locking mechanism (GPIO<sub>x</sub>\_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- 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

### 9.3 GPIO implementation

Table 43. GPIO implementation

Feature	STM32WB55xx	STM32WB35xx
Port A	0 to 15	0 to 15
Port B	0 to 15	0 to 9
Port C	0 to 15	14 to 15
Port D	0 to 15	-
Port E	0 to 4	4
Port H	0 to 1 and 3	3

## 9.4 GPIO functional description

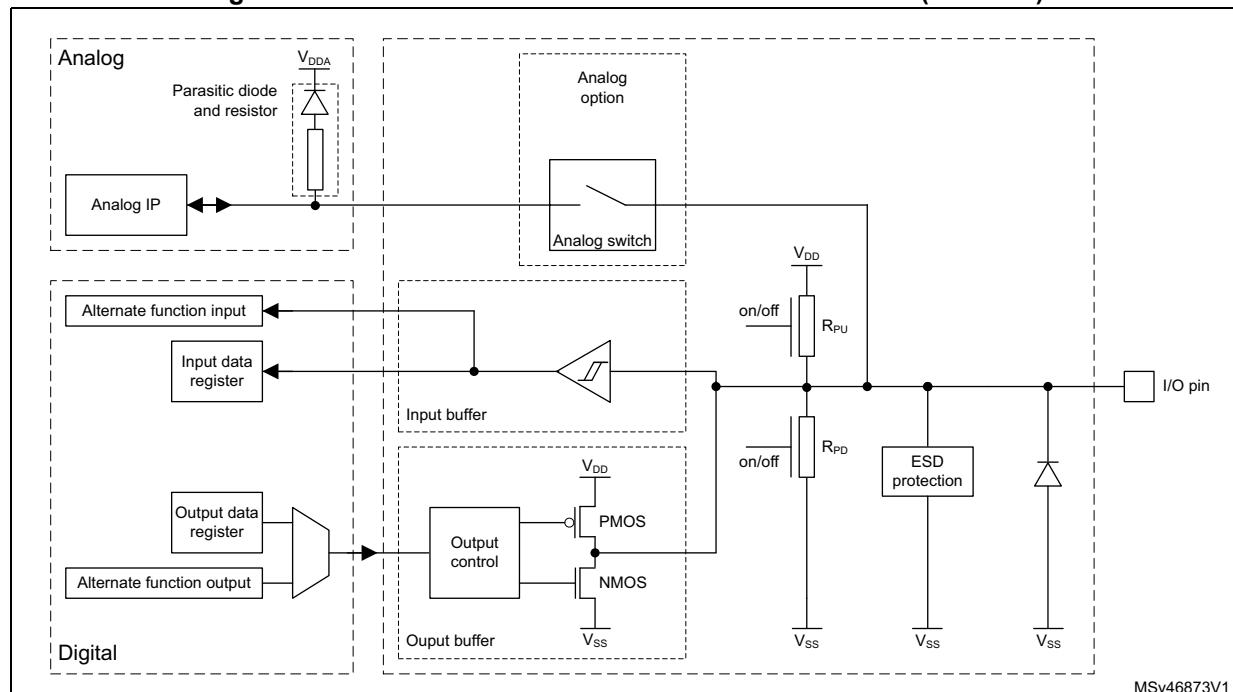
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

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

*Figure 20* shows the basic structure of a Three-volt tolerant (TT) and Five-volt tolerant (FT) I/O port bit.

**Figure 20. Three-volt or five-volt tolerant GPIO structure (TT or FT)**



**Note:** The parasitic diode in the analog domain is connected to  $V_{DDA}$  and cannot be used as a protection diode.

The voltage level called  $V_{DD\_FT}$  in some datasheets and reference manuals is inside the ESD protection block.

When the analog option is selected, the FT I/O is not five-volt tolerant anymore since the pin is supplied with  $V_{DDA}$ .

A TT or FT GPIO pin has no internal protection diode connected to supply ( $V_{DD}$ ). There is no physical limitation against over-voltage. Therefore, for applications requiring a limited voltage threshold, it is recommended to connect an external diode to  $V_{DD}$ .

*Table 44* gives the possible port bit configurations.

**Table 44. Port bit configuration table<sup>(1)</sup>**

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

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

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

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

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

- PA15: JTDI in input mode with pull-up
- PA14: JTCK/SWCLK in input mode with pull-down
- PA13: JTMS/SWDAT in input mode with pull-up
- PB4: NJTRST in input mode with pull-up
- PB3: JTDO in Hi-Z mode no pulls

PH3/BOOT0 is in input mode during the reset until at least the end of the option byte loading phase. See [Section 9.4.15: Using PH3 as GPIO](#).

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

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

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

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

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

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

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx\_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

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

To use an I/O in a given configuration, the user has to proceed as follows:

- **Debug function:** after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- **GPIO:** configure the desired I/O as output, input or analog in the GPIOx\_MODER register.
- **Peripheral alternate function:**
  - Connect the I/O to the desired AFx in one of the GPIOx\_AFRL or GPIOx\_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx\_OTYPER, GPIOx\_PUPDR and GPIOx\_OSPEEDER registers, respectively.

- Configure the desired I/O as an alternate function in the GPIOx\_MODER register.
- **Additional functions:**
  - For the ADC, PVD and COMP, configure the desired I/O in analog mode in the GPIOx\_MODER register and configure the required function in the ADC and COMP registers.
  - For the additional functions like RTC, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

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

#### 9.4.3 I/O port control registers

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

#### 9.4.4 I/O port data registers

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

See [Section 9.5.5](#) and [Section 9.5.6](#) for the register descriptions.

#### 9.4.5 I/O data bitwise handling

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

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

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

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

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

#### 9.4.6 GPIO locking mechanism

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

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

The LOCK sequence (refer to [Section 9.5.8](#)) can only be performed using a word (32-bit long) access to the GPIO<sub>x</sub>\_LCKR register due to the fact that GPIO<sub>x</sub>\_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in [Section 9.5.8](#).

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

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

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

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

#### 9.4.8 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 14: Extended interrupt and event controller \(EXTI\)](#) and to [Section 14.4: EXTI functional description](#).

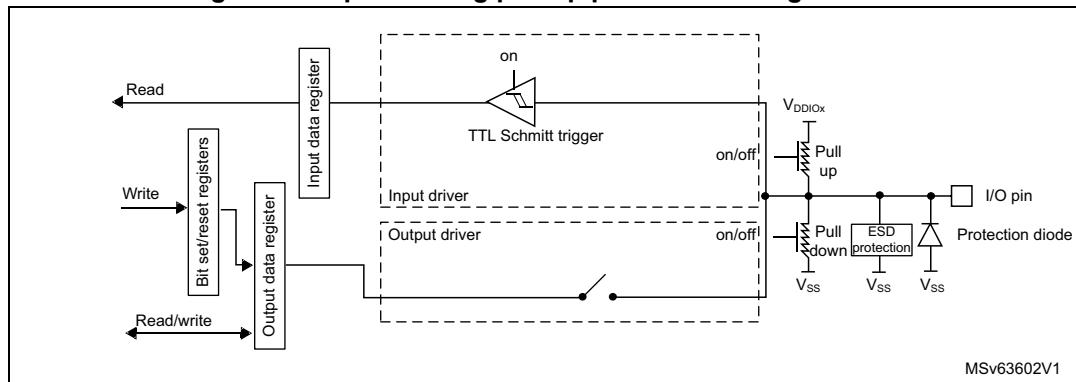
#### 9.4.9 Input configuration

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 GPIO<sub>x</sub>\_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register provides the I/O state

[Figure 21](#) shows the input configuration of the I/O port bit.

Figure 21. Input floating/pull up/pull down configurations



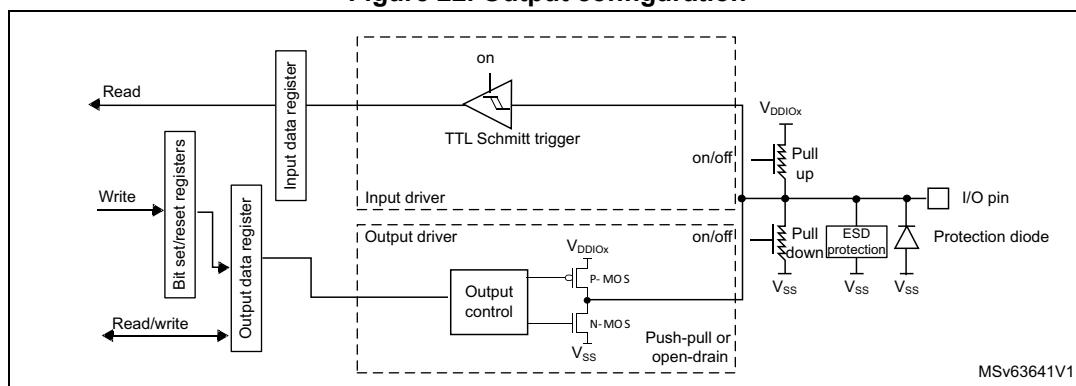
#### 9.4.10 Output configuration

When the I/O port is programmed as output:

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

*Figure 22* shows the output configuration of the I/O port bit.

Figure 22. Output configuration



## 9.4.11 Alternate function configuration

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

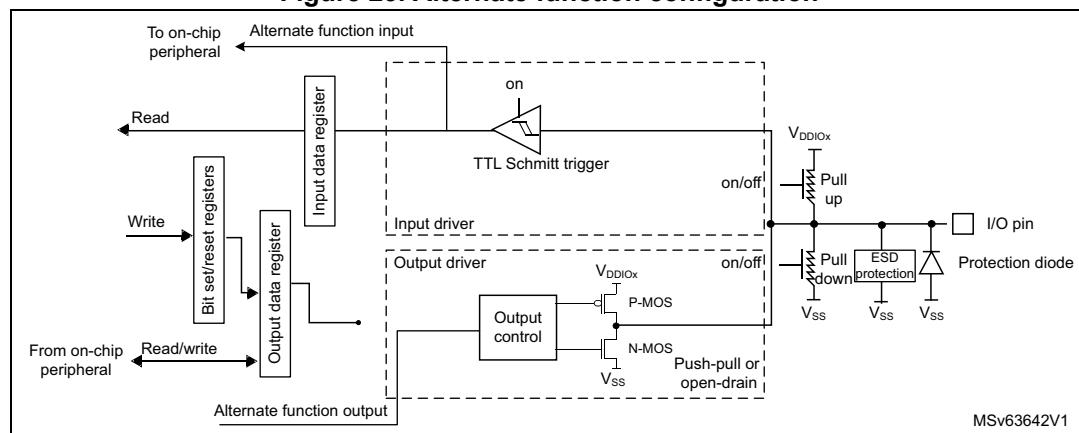
- The output buffer can be configured in open-drain or push-pull mode
  - The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
  - The Schmitt trigger input is activated
  - The weak pull-up and pull-down resistors are activated or not depending on the value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

*Note:*

The alternate function configuration described above is not applied when the selected alternate function is an LCD function. In this case, the I/O, programmed as an alternate function output, is configured as described in the analog configuration.

Figure 23 shows the Alternate function configuration of the I/O port bit.

**Figure 23. Alternate function configuration**



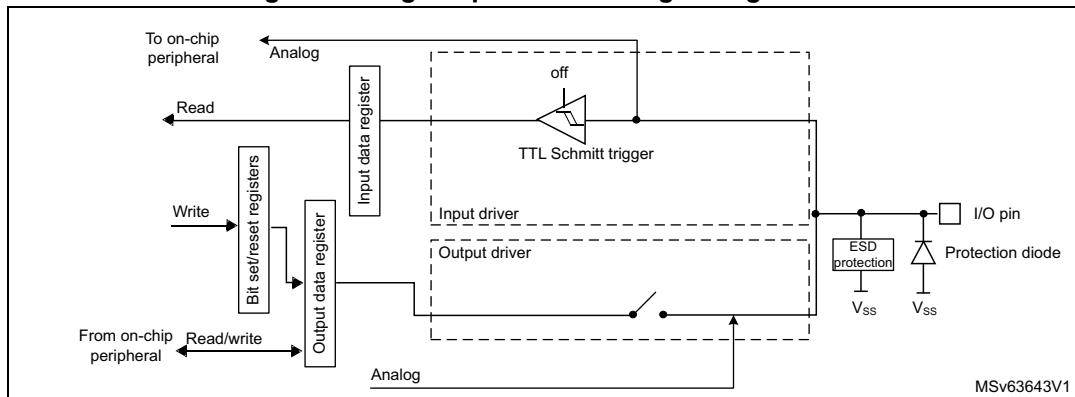
### 9.4.12 Analog configuration

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

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

Figure 24 shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 24. High impedance-analog configuration



#### 9.4.13 Using the LSE oscillator pins as GPIOs

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

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

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

*Note:* *The HSE OSC\_IN and OSC\_OUT pins are dedicated oscillator pins and cannot be used as GPIO.*

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

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

For details about I/O control by the RTC, refer to [Section 29.4: RTC functional description](#).

#### 9.4.15 Using PH3 as GPIO

PH3 may be used as boot pin (BOOT0) or as a GPIO. Depending on the nSWBOOT0 bit in the user option byte, it switches from the input mode to the analog input mode:

- After the option byte loading phase if nSWBOOT0 = 1.
- After reset if nSWBOOT0 = 0.

## 9.5 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to [Table 45](#).

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

### 9.5.1 GPIO port mode register (GPIOx\_MODER) (x =A to E and H)

Address offset:0x00

Reset value: 0xABFF FFFF (for port A)

Reset value: 0xFFFF FEBF (for port B)

Reset value: 0xFFFF FFFF for port C, D

Reset value: 0x0000 03FF for port E

Reset value: 0x0000 00CF for port H

Port E[31:10] are reserved

Port H[31:8, 5:4] are reserved

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw										

Bits 31:0 **MODE[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)

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

00: Input mode

01: General purpose output mode

10: Alternate function mode

11: Analog mode (reset state)

### 9.5.2 GPIO port output type register (GPIOx\_OTYPER) (x = A to E and H)

Address offset: 0x04

Reset value: 0x0000 0000

Port E[31:5] are reserved

Port H[31:4, 2] are reserved

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

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

Bits 15:0 **OT[15:0]**: Port x configuration I/O pin y (y = 15 to 0)

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

0: Output push-pull (reset state)

1: Output open-drain

### 9.5.3 GPIO port output speed register (GPIOx\_OSPEEDR) (x = A to E and H)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 00C0 (for port B)

Reset value: 0x0000 0000 (for other ports)

Port E[31:10] are reserved

Port H[31:8, 5:4] are reserved

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






















































































































































































































































































































































































































































































<tbl\_r cells="16" ix="2

Bits 31:0 **OSPEED[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)

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

- 00: Low speed
- 01: Medium speed
- 10: Fast speed
- 11: High speed

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

#### 9.5.4 GPIO port pull-up/pull-down register (GPIOx\_PUPDR) (x = A to E and H)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for other ports)

Port E[31:10] are reserved

Port H[31:8, 5:4] are reserved

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw										

Bits 31:0 **PUPD[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)

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

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

#### 9.5.5 GPIO port input data register (GPIOx\_IDR) (x = A to E and H)

Address offset: 0x10

Reset value: 0x0000 XXXX

Port E[31:5] are reserved

Port H[31:4, 2] are reserved

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **ID[15:0]**: Port x input data I/O pin y (y = 15 to 0)

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

### 9.5.6 GPIO port output data register (GPIOx\_ODR) (x = A to E and H)

Address offset: 0x14

Reset value: 0x0000 0000

Port E[31:5] are reserved

Port H[31:4, 2] are reserved

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw															

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

Bits 15:0 **OD[15:0]**: Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

*Note: For atomic bit set/reset, OD bits can be individually set and/or reset by writing to GPIO port bit set/reset register (GPIOx\_BSRR) (x = A to E and H).*

### 9.5.7 GPIO port bit set/reset register (GPIOx\_BSRR) (x = A to E and H)

Address offset: 0x18

Reset value: 0x0000 0000

Port E[31:21, 15:5] are reserved

Port H[31:20, 18, 15:4, 2] are reserved

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BR[15:0]**: Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Resets the corresponding ODx bit

*Note: If both BSx and BRx are set, BSx has priority.*

Bits 15:0 **BS[15:0]**: Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Sets the corresponding ODx bit

## 9.5.8 GPIO port configuration lock register (GPIOx\_LCKR) (x = A to E and H)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

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

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

Address offset: 0x1C

Reset value: 0x0000 0000

Port E[31:17, 15:5] are reserved

Port H[31:17, 15:4, 2] are reserved

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 16 **LCKK**: Lock key

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

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx\_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

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

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

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

RD LCKR

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

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

*Any error in the lock sequence aborts the lock.*

*After the first lock sequence on any bit of the port, any read access on the LCKK bit returns '1' until the next MCU reset or peripheral reset.*

Bits 15:0 **LCK[15:0]**: Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is '0'.

0: Port configuration not locked

1: Port configuration locked

### 9.5.9 GPIO alternate function low register (GPIOx\_AFRL) (x = A to E and H)

Address offset: 0x20

Reset value: 0x0000 0000

Port E[31:20] are reserved

Port H[31:16, 11:8] are reserved

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw												

Bits 31:0 **AFSEL[7:0][3:0]**: Alternate function selection for port x I/O pin y (y = 7 to 0)

These bits are written by software to configure alternate function I/Os.

0000: AF0  
0001: AF1  
0010: AF2  
0011: AF3  
0100: AF4  
0101: AF5  
0110: AF6  
0111: AF7  
1000: AF8  
1001: AF9  
1010: AF10  
1011: AF11  
1100: AF12  
1101: AF13  
1110: AF14  
1111: AF15

### 9.5.10 GPIO alternate function high register (GPIOx\_AFRH) (x = A to E and H)

Address offset: 0x24

Reset value: 0x0000 0000

Port E[31:0] are reserved

Port H[31:0] are reserved

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw												

Bits 31:0 **AFSEL[15:8][3:0]**: Alternate function selection for port x I/O pin y (y = 15 to 8)

These bits are written by software to configure alternate function I/Os.

0000: AF0  
0001: AF1  
0010: AF2  
0011: AF3  
0100: AF4  
0101: AF5  
0110: AF6  
0111: AF7  
1000: AF8  
1001: AF9  
1010: AF10  
1011: AF11  
1100: AF12  
1101: AF13  
1110: AF14  
1111: AF15

### 9.5.11 GPIO port bit reset register (GPIOx\_BRR) (x = A to E and H)

Address offset: 0x28

Reset value: 0x0000 0000

Port E[31:5] are reserved

Port H[31:4, 2] are reserved

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bits 15:0 **BR[15:0]**: Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Reset the corresponding ODx bit

### 9.5.12 GPIO register map

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

**Table 45. GPIO register map and reset values**

Offset	Register name		
0x00	GPIOA_MODER	1	MODE15[1:0]
		0	MODE15[1:0]
0x00	GPIOB_MODER	1	MODE15[1:0]
		0	MODE15[1:0]
0x00	GPIO <sub>x</sub> _MODER (where x = C, D)	1	MODE15[1:0]
		0	MODE15[1:0]
0x00	GPIOE_MODER	1	MODE15[1:0]
		0	MODE15[1:0]
0x00	GPIOH_MODER	1	MODE15[1:0]
		0	MODE15[1:0]
0x04	GPIO <sub>x</sub> _OTYPER (where x = A..D)	1	MODE15[1:0]
		0	MODE15[1:0]
0x04	GPIOE_OTYPER	1	MODE15[1:0]
		0	MODE15[1:0]
0x04	GPIOH_OTYPER	1	MODE15[1:0]
		0	MODE15[1:0]
0x08	GPIOA_OSPEEDR	0	OSPEED15[1:0]
		0	OSPEED15[1:0]
0x08	GPIOB_OSPEEDR	0	OSPEED15[1:0]
		0	OSPEED15[1:0]
0x08	GPIO <sub>x</sub> _OSPEEDR	0	OSPEED15[1:0]
		0	OSPEED15[1:0]
0x08	GPIOE_OSPEEDR	0	OSPEED15[1:0]
		0	OSPEED15[1:0]
0x08	GPIOH_OSPEEDR	0	OSPEED15[1:0]
		0	OSPEED15[1:0]

**Table 45. GPIO register map and reset values (continued)**

**Table 45. GPIO register map and reset values (continued)**

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 10 System configuration controller (SYSCFG)

### 10.1 SYSCFG main features

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

- Remapping memory areas
- Managing the external interrupt line connection to the GPIOs
- Managing robustness feature
- Setting SRAM2 and PKA RAM write protection and software erase
- Configuring FPU interrupts
- Enabling /disabling I<sup>2</sup>C Fast-mode Plus driving capability on some I/Os and voltage booster for I/Os analog switches
- Interrupt pre-masking

### 10.2 SYSCFG registers

#### 10.2.1 SYSCFG memory remap register (SYSCFG\_MEMRMP)

This register is used for specific configurations on memory remap.

Address offset: 0x000

Reset value: 0x0000 000X (X is the memory mode selected by the BOOT0 pin and BOOT1 option bit)

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

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

Bits 2:0 **MEM\_MODE[2:0]**: Memory mapping selection

These bits control the memory internal mapping at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT mode setting. After reset these bits take the value selected by BOOT0 (pin or option bit depending on NSWBOOT0 option bit) and BOOT1 option bit.

000: Main Flash memory mapped at CPU1 0x00000000

001: System Flash memory mapped at CPU1 0x00000000

010: Reserved

011: SRAM1 mapped at CPU1 0x00000000

100: Reserved

101: Reserved

110: QUADSPI memory mapped at CPU1 0x00000000

111: Reserved



Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **BOOSTEN**: I/O analog switch voltage booster enable

0: I/O analog switches are supplied by  $V_{DDA}$  voltage. This is the recommended configuration when using the ADC in high  $V_{DDA}$  voltage operation.

1: I/O analog switches are supplied by a dedicated voltage booster (supplied by  $V_{DD}$ ). This is the recommended configuration when using the ADC in low  $V_{DDA}$  voltage operation.

Bits 7:0 Reserved, must be kept at reset value.

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

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EXTI3[2:0]			Res.	EXTI2[2:0]			Res.	EXTI1[2:0]			Res.	EXTI0[2:0]		
	rw	rw	rw												

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

Bits 14:12 **EXTI3[2:0]**: EXTI 3 configuration bits

These bits are written by software to select the source input for the EXTI3 external interrupt.

000: PA[3] pin

001: PB[3] pin

010: PC[3] pin (STM32WB55xx only)

011: PD[3] pin (STM32WB55xx only)

100: PE[3] pin (STM32WB55xx only)

101: Reserved

110: Reserved

111: PH[3] pin

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI2[2:0]**: EXTI 2 configuration bits

These bits are written by software to select the source input for the EXTI2 external interrupt.

000: PA[2] pin

001: PB[2] pin

010: PC[2] pin (STM32WB55xx only)

011: PD[2] pin (STM32WB55xx only)

100: PE[2] pin (STM32WB55xx only)

101: Reserved

110: Reserved

111: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI1[2:0]**: EXTI 1 configuration bits

These bits are written by software to select the source input for the EXTI1 external interrupt.

- 000: PA[1] pin
- 001: PB[1] pin
- 010: PC[1] pin (STM32WB55xx only)
- 011: PD[1] pin (STM32WB55xx only)
- 100: PE[1] pin (STM32WB55xx only)
- 101: Reserved
- 110: Reserved
- 111: PH[1] pin (STM32WB55xx only)

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI0[2:0]**: EXTI 0 configuration bits

These bits are written by software to select the source input for the EXTI0 external interrupt.

- 000: PA[0] pin
- 001: PB[0] pin
- 010: PC[0] pin (STM32WB55xx only)
- 011: PD[0] pin (STM32WB55xx only)
- 100: PE[0] pin (STM32WB55xx only)
- 101: Reserved
- 110: Reserved
- 111: PH[0] pin (STM32WB55xx only)

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

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

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EXTI7[2:0]			Res.	EXTI6[2:0]			Res.	EXTI5[2:0]			Res.	EXTI4[2:0]		
	rw	rw	rw												

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

Bits 14:12 **EXTI7[2:0]**: EXTI 7 configuration bits

These bits are written by software to select the source input for the EXTI7 external interrupt.

- 000: PA[7] pin
- 001: PB[7] pin
- 010: PC[7] pin (STM32WB55xx only)
- 011: PD[7] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI6[2:0]**: EXTI 6 configuration bits

These bits are written by software to select the source input for the EXTI6 external interrupt.

- 000: PA[6] pin
- 001: PB[6] pin
- 010: PC[6] pin (STM32WB55xx only)
- 011: PD[6] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI5[2:0]**: EXTI 5 configuration bits

These bits are written by software to select the source input for the EXTI5 external interrupt.

- 000: PA[5] pin
- 001: PB[5] pin
- 010: PC[5] pin (STM32WB55xx only)
- 011: PD[5] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI4[2:0]**: EXTI 4 configuration bits

These bits are written by software to select the source input for the EXTI4 external interrupt.

- 000: PA[4] pin
- 001: PB[4] pin
- 010: PC[4] pin (STM32WB55xx only)
- 011: PD[4] pin (STM32WB55xx only)
- 100: PE[4] pin
- 101: Reserved
- 110: Reserved
- 111: Reserved

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

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

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EXTI11[2:0]			Res.	EXTI10[2:0]			Res.	EXTI9[2:0]			Res.	EXTI8[2:0]		
rw	rw	rw			rw	rw	rw		rw	rw	rw		rw	rw	rw

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

Bits 14:12 **EXTI11[2:0]**: EXTI 11 configuration bits

These bits are written by software to select the source input for the EXTI11 external interrupt.

- 000: PA[11] pin
- 001: PB[11] pin (STM32WB55xx only)
- 010: PC[11] pin (STM32WB55xx only)
- 011: PD[11] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI10[2:0]**: EXTI 10 configuration bits

These bits are written by software to select the source input for the EXTI10 external interrupt.

- 000: PA[10] pin
- 001: PB[10] pin (STM32WB55xx only)
- 010: PC[10] pin (STM32WB55xx only)
- 011: PD[10] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI9[2:0]**: EXTI 9 configuration bits

These bits are written by software to select the source input for the EXTI9 external interrupt.

- 000: PA[9] pin
- 001: PB[9] pin
- 010: PC[9] pin (STM32WB55xx only)
- 011: PD[9] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI8[2:0]**: EXTI 8 configuration bits

These bits are written by software to select the source input for the EXTI8 external interrupt.

- 000: PA[8] pin
- 001: PB[8] pin
- 010: PC[8] pin (STM32WB55xx only)
- 011: PD[8] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

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

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

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EXTI15[2:0]			Res.	EXTI14[2:0]			Res.	EXTI13[2:0]			Res.	EXTI12[2:0]		
rw	rw	rw			rw	rw	rw		rw	rw	rw		rw	rw	rw

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

Bits 14:12 **EXTI15[2:0]**: EXTI15 configuration bits

These bits are written by software to select the source input for the EXTI15 external interrupt.

- 000: PA[15] pin
- 001: PB[15] pin (STM32WB55xx only)
- 010: PC[15] pin
- 011: PD[15] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI14[2:0]**: EXTI14 configuration bits

These bits are written by software to select the source input for the EXTI14 external interrupt.

- 000: PA[14] pin
- 001: PB[14] pin (STM32WB55xx only)
- 010: PC[14] pin
- 011: PD[14] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EXTI13[2:0]**: EXTI13 configuration bits

These bits are written by software to select the source input for the EXTI13 external interrupt.

- 000: PA[13] pin
- 001: PB[13] pin (STM32WB55xx only)
- 010: PC[13] pin (STM32WB55xx only)
- 011: PD[13] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EXTI12[2:0]**: EXTI12 configuration bits

These bits are written by software to select the source input for the EXTI12 external interrupt.

- 000: PA[12] pin
- 001: PB[12] pin (STM32WB55xx only)
- 010: PC[12] pin (STM32WB55xx only)
- 011: PD[12] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

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

### 10.2.7 SYSCFG SRAM2 control and status register (SYSCFG\_SCSR)

Address offset: 0x18

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C2RFD	Res.	Res.													
<b>rw</b>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM2 BSY	SRAM2 ER
														r	rw

Bit 31 **C2RFD**: CPU2 SRAM fetch (execution) disable.

This bit can be set by software and be reset by a hardware reset, including a reset after Standby. Software writing 'b0 has no effect.

0: CPU2 fetch from SRAM1, SRAM2a and SRAM2b enabled, allows CPU2 to fetch and execute code from SRAMs

1: CPU2 fetch from SRAM1, SRAM2a and SRAM2b disabled, Any CPU2 fetch from SRAMs generates a bus error.

Bits 30:2 Reserved, must be kept at reset value.

Bit 1 **SRAM2BSY**: SRAM2 and PKA RAM busy by erase operation

0: Nor SRAM2 neither PKA RAM erase operation is on going.

1: SRAM2 and/or PKA RAM erase operation is on going.

Bit 0 **SRAM2ER**: SRAM2 and PKA RAM Erase

Setting this bit starts a hardware SRAM2 and PKA RAM erase operation. This bit is automatically cleared at the end of the SRAM2 and PKA RAM erase operation.

*Note: This bit is write-protected: setting this bit is possible only after the correct key sequence is written in the SYSCFG\_SKR register.*

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

Address offset: 0x01C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPF	Res.	Res.	Res.	Res.	ECCL	PVDL	SPL	CLL						
						rc_w1					rs	rs	rs	rs	

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

Bit 8 **SPF**: SRAM2 parity error flag

This bit is set by hardware when an SRAM2 parity error is detected. It is cleared by software by writing '1'.

0: No SRAM2 parity error detected

1: SRAM2 parity error detected

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

Bit 3 **ECCL**: ECC Lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the Flash ECC error connection to TIM1/16/17 Break input.

0: ECC error disconnected from TIM1/16/17 Break input.

1: ECC error connected to TIM1/16/17 Break input.

Bit 2 **PVDL**: PVD lock enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM1/16/17 Break input, as well as the PVDE and PLS[2:0] in the PWR\_CR2 register.

0: PVD interrupt disconnected from TIM1/16/17 Break input. PVDE and PLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM1/16/17 Break input, PVDE and PLS[2:0] bits are read only.

Bit 1 **SPL**: SRAM2 parity lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM2 parity error signal connection to TIM1/16/17 Break inputs.

0: SRAM2 parity error signal disconnected from TIM1/16/17 Break inputs

1: SRAM2 parity error signal connected to TIM1/16/17 Break inputs

Bit 0 **CLL**: CPU1 LOCKUP (Hardfault) output enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the connection of CPU1 LOCKUP (Hardfault) output to TIM1/16/17 Break input

0: CPU1 LOCKUP output disconnected from TIM1/16/17 Break inputs

1: CPU1 LOCKUP output connected to TIM1/16/17 Break inputs

### 10.2.9 SYSCFG SRAM2 write protection register (SYSCFG\_SWPR1)

Address offset: 0x020

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP
rs															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
rs															

Bits 31:0 **PxWP** (x= 0 to 31): SRAM2 1Kbyte page x write protection

These bits are set by software and cleared only by a system reset.

0: Write protection of SRAM2 1Kbyte page x is disabled.

1: Write protection of SRAM2 1Kbyte page x is enabled.

### 10.2.10 SYSCFG SRAM2 key register (SYSCFG\_SKR)

Address offset: 0x024

System reset value: 0x0000 0000

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

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

Bits 7:0 **KEY[7:0]**: SRAM2 write protection key for software erase

The following steps are required to unlock the write protection of the SRAM2ER bit in the SYSCFG\_CFGR2 register.

1. Write 0xCA into Key[7:0]

2. Write 0x53 into Key[7:0]

Writing a wrong key reactivates the write protection.

### 10.2.11 SYSCFG SRAM2 write protection register 2 (SYSCFG\_SWPR2)

Address offset: 0x028

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP
rs															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP
rs															

Bits 31:0 **PxWP** (x= 32 to 63): SRAM2 1 Kbyte page x write protection

These bits are set by software and cleared only by a system reset.

0: Write protection of SRAM2 1 Kbyte page x is disabled.

1: Write protection of SRAM2 1 Kbyte page x is enabled.

### 10.2.12 SYSCFG CPU1 interrupt mask register 1 (SYSCFG\_IMR1)

Address offset: 0x100

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI15 IM	EXTI14 IM	EXTI13 IM	EXTI12 IM	EXTI11 IM	EXTI10 IM	EXTI9 IM	EXTI8 IM	EXTI7 IM	EXTI6 IM	EXTI5 IM	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM17 IM	TIM16 IM	TIM1 IM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

Bits 31:21 **xxxIM**: Peripheral xxx interrupt mask to CPU1

0: Peripheral xxx interrupt forwarded to CPU1

1: Peripheral xxx interrupt to CPU1 masked.

Bits 20:16 Reserved, must be kept at reset value.

Bits 15:13 **xxxIM**: Peripheral xxx interrupt mask to CPU1

0: Peripheral xxx interrupt forwarded to CPU1

1: Peripheral xxx interrupt to CPU1 masked.

Bits 12:0 Reserved, must be kept at reset value.

### 10.2.13 SYSCFG CPU1 interrupt mask register 2 (SYSCFG\_IMR2)

Address offset: 0x104

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PVDIM	Res.	PVM3IM	Res.	PVM1IM										
											rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.											

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

Bit 20 **xxxIM**: Peripheral xxx interrupt mask to CPU1

- 0: Peripheral xxx interrupt forwarded to CPU1
- 1. Peripheral xxx interrupt to CPU1 masked.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **xxxIM**: Peripheral xxx interrupt mask to CPU1

- 0: Peripheral xxx interrupt forwarded to CPU1
- 1. Peripheral xxx interrupt to CPU1 masked.

Bit 17 Reserved, must be kept at reset value.

Bits 16 **xxxIM**: Peripheral xxx interrupt mask to CPU1

- 0: Peripheral xxx interrupt forwarded to CPU1
- 1. Peripheral xxx interrupt to CPU1 masked.

Bits 15:0 Reserved, must be kept at reset value.

### 10.2.14 SYSCFG CPU2 interrupt mask register 1 (SYSCFG\_C2IMR1)

Address offset: 0x108

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI15 IM	EXTI14 IM	EXTI13 IM	EXTI12 IM	EXTI11 IM	EXTI10 IM	EXTI9 IM	EXTI8 IM	EXTI7 IM	EXTI6 IM	EXTI5 IM	EXTI4 IM	EXTI3 IM	EXTI2 IM	EXTI1 IM	EXTI0 IM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ADC IM	COMP IM	AES1 IM	RNG IM	PKA IM	Res.	FLASH IM	RCC IM	RTC ALARM IM	RTC WKUP IM	Res.	Res.	RTC STAMP TAMP LSECSS IM
			rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:16 **xxxIM**: Peripheral xxx interrupt mask to CPU2

- 0: Peripheral xxx interrupt forwarded to CPU2
- 1. Peripheral xxx interrupt to CPU2 masked.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **xxxIM**: Peripheral xxx interrupt mask to CPU2

- 0: Peripheral xxx interrupt forwarded to CPU2
- 1. Peripheral xxx interrupt to CPU2 masked.

Bit 7 Reserved, must be kept at reset value.

Bits 6:3 **xxxIM**: Peripheral xxx interrupt mask to CPU2

- 0: Peripheral xxx interrupt forwarded to CPU2
- 1. Peripheral xxx interrupt to CPU2 masked.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **xxxIM**: Peripheral xxx interrupt mask to CPU2

- 0: Peripheral xxx interrupt forwarded to CPU2
- 1. Peripheral xxx interrupt to CPU2 masked.

### 10.2.15 SYSCFG CPU2 interrupt mask register 2 (SYSCFG\_C2IMR2)

Address offset: 0x10C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	LCD IM	TSC IM	PVD IM	Res.	PVM3 IM	Res.	PVM1 IM							
									rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA MUX1 IM	DMA2 CH7 IM	DMA2 CH6 IM	DMA2 CH5 IM	DMA2 CH4 IM	DMA2 CH3 IM	DMA2 CH2 IM	DMA2 CH1 IM	Res.	DMA1 CH7 IM	DMA1 CH6 IM	DMA1 CH5 IM	DMA1 CH4 IM	DMA1 CH3 IM	DMA1 CH2 IM	DMA1 CH1 IM
rw		rw													

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

Bits 22:20 **xxxIM**: Peripheral xxx interrupt mask to CPU2

0: Peripheral xxx interrupt forwarded to CPU2

1. Peripheral xxx interrupt to CPU2 masked.

Note that bits 22:21 are reserved in STM32WB35xx devices.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **xxxIM**: Peripheral xxx interrupt mask to CPU2

0: Peripheral xxx interrupt forwarded to CPU2

1. Peripheral xxx interrupt to CPU2 masked.

Bit 17 Reserved, must be kept at reset value.

Bits 16:8 **xxxIM**: Peripheral xxx interrupt mask to CPU2

0: Peripheral xxx interrupt forwarded to CPU2

1. Peripheral xxx interrupt to CPU2 masked.

Bit7 Reserved, must be kept at reset value.

Bits 6:0 **xxxIM**: Peripheral xxx interrupt mask to CPU2

0: Peripheral xxx interrupt forwarded to CPU2

1. Peripheral xxx interrupt to CPU2 masked.

### 10.2.16 SYSCFG secure IP control register (SYSCFG\_SIPCR)

Address offset: 0x110

System reset value: 0x0000 0000

This register provides write access security and can only be written by the CPU2. A write access from the CPU1 is ignored and a bus error is generated. On any read access the register value is returned.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	SRNG	SPKA	SAES2	SAES1												
													rw	rw	rw	rw

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

Bit 3 **SRNG**: Enable true RNG security

0: True RNG security disabled (RNG registers can be accessed by both CPU1 and CPU2)

1: True RNG security enabled (RNG registers can only be accessed by the CPU2. Write accesses by the CPU1 generate a bus error, read access return 0 value).

Bit 2 **SPKA**: Enable PKA security

0: PKA security disabled (PKA registers can be accessed by both CPU1 and CPU2)

1: PKA security enabled. PKA registers can only be accessed by the CPU2. Write accesses by the CPU1 generate a bus error, read access return 0 value. When this bit is set and a system reset occurs (including a reset due to wakeup from Standby) the PKA RAM is erased.

Bit 1 **SAES2**: Enable AES2 security

0: AES2 security disabled (AES2 registers can be accessed by both CPU1 and CPU2)

1: AES2 security enabled. (AES2 registers can only be accessed by the CPU2. Write accesses by the CPU1 generate a bus error, read access return 0 value).

Bit 0 **SAES1**: Enable AES1 KEY[7:0] security

0: AES1 KEY[7:0] security disabled (AES1 KEY[7:0] registers can be accessed by both CPU1 and CPU2)

1: AES1 KEY[7:0] security enabled (AES1 KEY[7:0] registers can only be accessed by the CPU2. Write accesses by the CPU1 generate a bus error, read access return zero value).

### 10.2.17 SYSCFG register map

The following table summarizes the SYSCFG register map and the reset values.

Table 46. SYSCFG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
0x000	SYSCFG_MEMRMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x004	SYSCFG_CFGR1	FPU_IE[5..0]					0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x008	SYSCFG_EXTICR1	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
0x00C	SYSCFG_EXTICR2	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
0x010	SYSCFG_EXTICR3	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
0x014	SYSCFG_EXTICR4	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
0x018	SYSCFG_SCSR	Reset value	0	C2RFD	Res.																																			
0x01C	SYSCFG_CFGR2	Reset value	0	P31WP	Res.																																			
0x020	SYSCFG_SWPR	Reset value	0	P32WP	Res.																																			
0x024	SYSCFG_SKR	Reset value	0	P33WP	Res.																																			
0x028	SYSCFG_SWPR2	Reset value	0	P34WP	Res.																																			

Table 46. SYSCFG register map and reset values (continued)

Offset	Register	Reset value
0x100	SYSCFG_IMR1	31
	Reset value	0 EXTI15IM 0 EXTI14IM 0 EXTI13IM 0 EXTI12IM 0 EXTI11IM 0 EXTI10IM 0 EXTI9IM 0 EXTI8IM 0 EXTI7IM 0 EXTI6IM 0 EXTI5IM 0 PVDIM 0 PVM3IM 0 PVM2IM 0 PVM1IM
0x104	SYSCFG_IMR2	30
	Reset value	0 EXTI15IM 0 EXTI14IM 0 EXTI13IM 0 EXTI12IM 0 EXTI11IM 0 EXTI10IM 0 EXTI9IM 0 EXTI8IM 0 EXTI7IM 0 EXTI6IM 0 EXTI5IM 0 PVDIM 0 PVM3IM 0 PVM2IM 0 PVM1IM
0x108	SYSCFG_C2IMR1	29
	Reset value	0 EXTI15IM 0 EXTI14IM 0 EXTI13IM 0 EXTI12IM 0 EXTI11IM 0 EXTI10IM 0 EXTI9IM 0 EXTI8IM 0 EXTI7IM 0 EXTI6IM 0 EXTI5IM 0 PVDIM 0 PVM3IM 0 PVM2IM 0 PVM1IM
0x10C	SYSCFG_C2IMR2	28
	Reset value	0 LCDIM 0 TSCIM 0 PVDIM 0 PVM3IM 0 PVM2IM 0 PVM1IM
0x110	SYSCFG_SIPCR	27
	Reset value	0 DMA1UX1IM 0 DMA2CH7IM 0 DMA2CH6IM 0 DMA2CH5IM 0 DMA2CH4IM 0 DMA2CH3IM 0 DMA2CH2IM 0 DMA2CH1IM 0 DMA1CH7IM 0 DMA1CH6IM 0 DMA1CH5IM 0 DMA1CH4IM 0 DMA1CH3IM 0 DMA1CH2IM 0 DMA1CH1IM 0 RTCSTAMPTAMPISECSSIM

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

# 11 Direct memory access controller (DMA)

## 11.1 Introduction

The direct memory access (DMA) controller is a bus master and system peripheral.

The DMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories, upon the control of an off-loaded CPU.

The DMA controller features a single AHB master architecture.

There are two instances of DMA, DMA1 and DMA2.

Each channel is dedicated to managing memory access requests from one or more peripherals. Each DMA includes an arbiter for handling the priority between DMA requests.

## 11.2 DMA main features

- Single AHB master
- Peripheral-to-memory, memory-to-peripheral, memory-to-memory and peripheral-to-peripheral data transfers
- Access, as source and destination, to on-chip memory-mapped devices such as Flash memory, SRAM, and AHB and APB peripherals
- All DMA channels independently configurable:
  - Each channel is associated either with a DMA request signal coming from a peripheral, or with a software trigger in memory-to-memory transfers. This configuration is done by software.
  - Priority between the requests is programmable by software (4 levels per channel: very high, high, medium, low) and by hardware in case of equality (such as request to channel 1 has priority over request to channel 2).
  - Transfer size of source and destination are independent (byte, half-word, word), emulating packing and unpacking. Source and destination addresses must be aligned on the data size.
  - Support of transfers from/to peripherals to/from memory with circular buffer management
  - Programmable number of data to be transferred: 0 to  $2^{16} - 1$
- Generation of an interrupt request per channel. Each interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.

## 11.3 DMA implementation

### 11.3.1 DMA1 and DMA2

DMA1 and DMA2 are implemented with the hardware configuration parameters shown in the table below.

**Table 47. DMA1 and DMA2 implementation**

Feature	DMA1	DMA2
Number of channels	7	7

### 11.3.2 DMA request mapping

The DMA controller is connected to DMA requests from the AHB/APB peripherals through the DMAMUX peripheral.

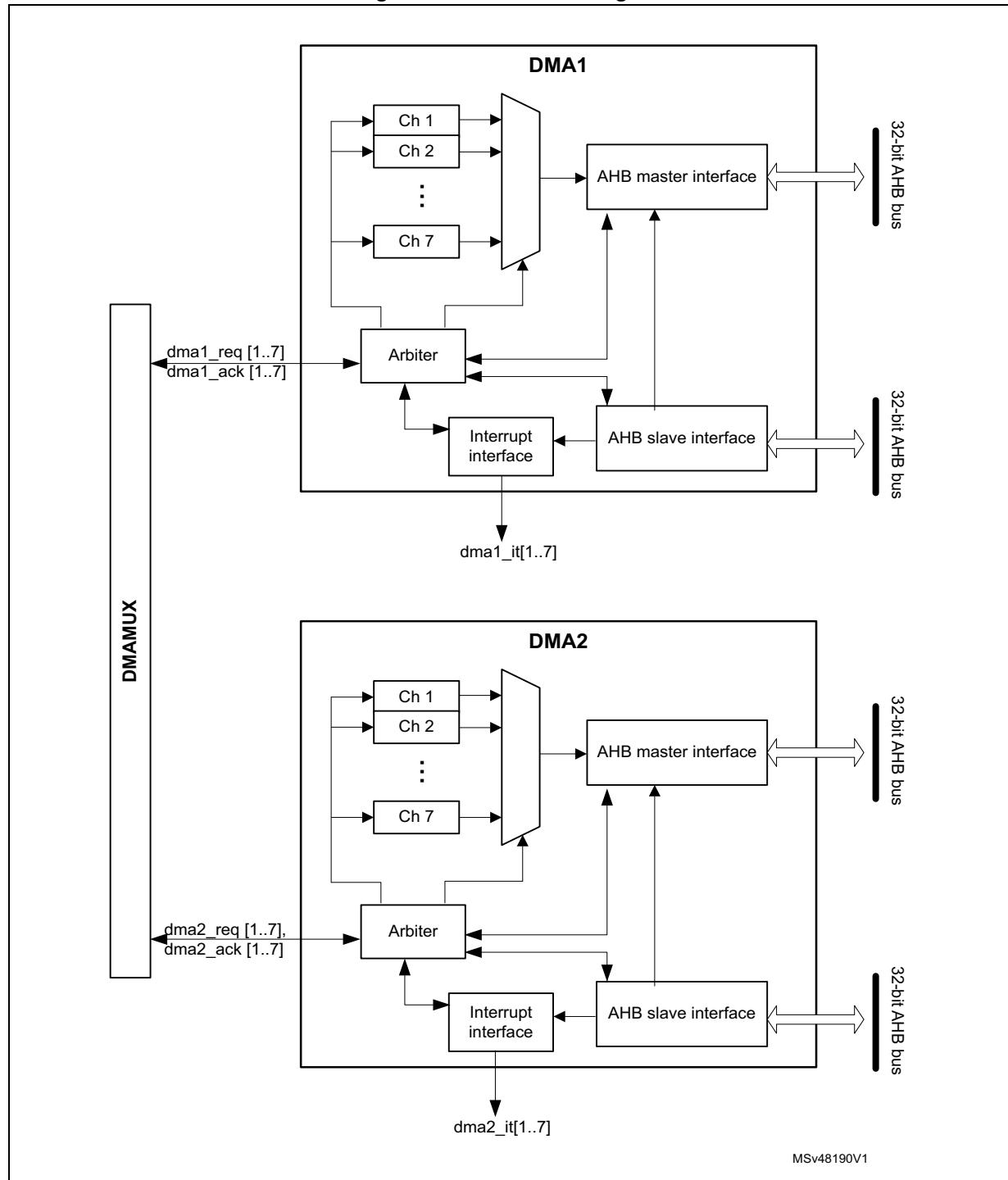
For the mapping of the different requests, refer to the DMAMUX section.

## 11.4 DMA functional description

### 11.4.1 DMA block diagram

The DMA block diagram is shown in the figure below.

Figure 25. DMA block diagram



The DMA controller performs direct memory transfer by sharing the AHB system bus with other system masters. The bus matrix implements round-robin scheduling. DMA requests

may stop the CPU access to the system bus for a number of bus cycles, when CPU and DMA target the same destination (memory or peripheral).

According to its configuration through the AHB slave interface, the DMA controller arbitrates between the DMA channels and their associated received requests. The DMA controller also schedules the DMA data transfers over the single AHB port master.

The DMA controller generates an interrupt per channel to the interrupt controller.

#### 11.4.2 DMA pins and internal signals

**Table 48. DMA internal input/output signals**

Signal name	Signal type	Description
dma_req[x]	Input	DMA channel x request
dma_ack[x]	Output	DMA channel x acknowledge
dma_it[x]	Output	DMA channel x interrupt

#### 11.4.3 DMA transfers

The software configures the DMA controller at channel level, in order to perform a block transfer, composed of a sequence of AHB bus transfers.

A DMA block transfer may be requested from a peripheral, or triggered by the software in case of memory-to-memory transfer.

After an event, the following steps of a single DMA transfer occur:

1. The peripheral sends a single DMA request signal to the DMA controller.
2. The DMA controller serves the request, depending on the priority of the channel associated to this peripheral request.
3. As soon as the DMA controller grants the peripheral, an acknowledge is sent to the peripheral by the DMA controller.
4. The peripheral releases its request as soon as it gets the acknowledge from the DMA controller.
5. Once the request is de-asserted by the peripheral, the DMA controller releases the acknowledge.

The peripheral may order a further single request and initiate another single DMA transfer.

The request/acknowledge protocol is used when a peripheral is either the source or the destination of the transfer. For example, in case of memory-to-peripheral transfer, the peripheral initiates the transfer by driving its single request signal to the DMA controller. The DMA controller reads then a single data in the memory and writes this data to the peripheral.

For a given channel x, a DMA block transfer consists of a repeated sequence of:

- a single DMA transfer, encapsulating two AHB transfers of a single data, over the DMA AHB bus master:
  - a single data read (byte, half-word or word) from the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.

The start address used for the first single transfer is the base address of the

peripheral or memory, and is programmed in the DMA\_CPARx or DMA\_CMARx register.

- a single data write (byte, half-word or word) to the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.

The start address used for the first transfer is the base address of the peripheral or memory, and is programmed in the DMA\_CPARx or DMA\_CMARx register.

- post-decrementing of the programmed DMA\_CNDTRx register  
This register contains the remaining number of data items to transfer (number of AHB 'read followed by write' transfers).

This sequence is repeated until DMA\_CNDTRx is null.

*Note:* *The AHB master bus source/destination address must be aligned with the programmed size of the transferred single data to the source/destination.*

#### 11.4.4 DMA arbitration

The DMA arbiter manages the priority between the different channels.

When an active channel x is granted by the arbiter (hardware requested or software triggered), a single DMA transfer is issued (such as a AHB 'read followed by write' transfer of a single data). Then, the arbiter considers again the set of active channels and selects the one with the highest priority.

The priorities are managed in two stages:

- software: priority of each channel is configured in the DMA\_CCRx register, to one of the four different levels:
  - very high
  - high
  - medium
  - low
- hardware: if two requests have the same software priority level, the channel with the lowest index gets priority. For example, channel 2 gets priority over channel 4.

When a channel x is programmed for a block transfer in memory-to-memory mode, re arbitration is considered between each single DMA transfer of this channel x. Whenever there is another concurrent active requested channel, the DMA arbiter automatically alternates and grants the other highest-priority requested channel, which may be of lower priority than the memory-to-memory channel.

#### 11.4.5 DMA channels

Each channel may handle a DMA transfer between a peripheral register located at a fixed address, and a memory address. The amount of data items to transfer is programmable. The register that contains the amount of data items to transfer is decremented after each transfer.

A DMA channel is programmed at block transfer level.

## Programmable data sizes

The transfer sizes of a single data (byte, half-word, or word) to the peripheral and memory are programmable through, respectively, the PSIZE[1:0] and MSIZE[1:0] fields of the DMA\_CCRx register.

## Pointer incrementation

The peripheral and memory pointers may be automatically incremented after each transfer, depending on the PINC and MINC bits of the DMA\_CCRx register.

If the **incremented mode** is enabled (PINC or MINC set to 1), the address of the next transfer is the address of the previous one incremented by 1, 2 or 4, depending on the data size defined in PSIZE[1:0] or MSIZE[1:0]. The first transfer address is the one programmed in the DMA\_CPARx or DMA\_CMARx register. During transfers, these registers keep the initially programmed value. The current transfer addresses (in the current internal peripheral/memory address register) are not accessible by software.

If the channel x is configured in **non-circular mode**, no DMA request is served after the last data transfer (once the number of single data to transfer reaches zero). The DMA channel must be disabled in order to reload a new number of data items into the DMA\_CNDTRx register.

**Note:** *If the channel x is disabled, the DMA registers are not reset. The DMA channel registers (DMA\_CCRx, DMA\_CPARx and DMA\_CMARx) retain the initial values programmed during the channel configuration phase.*

In **circular mode**, after the last data transfer, the DMA\_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address values from the DMA\_CPARx and DMA\_CMARx registers.

## Channel configuration procedure

The following sequence is needed to configure a DMA channel x:

1. Set the peripheral register address in the DMA\_CPARx register.  
The data is moved from/to this address to/from the memory after the peripheral event, or after the channel is enabled in memory-to-memory mode.
2. Set the memory address in the DMA\_CMARx register.  
The data is written to/read from the memory after the peripheral event or after the channel is enabled in memory-to-memory mode.
3. Configure the total number of data to transfer in the DMA\_CNDTRx register.  
After each data transfer, this value is decremented.
4. Configure the parameters listed below in the DMA\_CCRx register:
  - the channel priority
  - the data transfer direction
  - the circular mode
  - the peripheral and memory incremented mode
  - the peripheral and memory data size
  - the interrupt enable at half and/or full transfer and/or transfer error
5. Activate the channel by setting the EN bit in the DMA\_CCRx register.

A channel, as soon as enabled, may serve any DMA request from the peripheral connected to this channel, or may start a memory-to-memory block transfer.

**Note:** *The two last steps of the channel configuration procedure may be merged into a single access to the DMA\_CCRx register, to configure and enable the channel.*

### Channel state and disabling a channel

A channel x in active state is an enabled channel (read DMA\_CCRx.EN = 1). An active channel x is a channel that must have been enabled by the software (DMA\_CCRx.EN set to 1) and afterwards with no occurred transfer error (DMA\_ISR.TEIFx = 0). In case there is a transfer error, the channel is automatically disabled by hardware (DMA\_CCRx.EN = 0).

The three following use cases may happen:

- Suspend and resume a channel

This corresponds to the two following actions:

- An active channel is disabled by software (writing DMA\_CCRx.EN = 0 whereas DMA\_CCRx.EN = 1).
- The software enables the channel again (DMA\_CCRx.EN set to 1) without reconfiguring the other channel registers (such as DMA\_CNDTRx, DMA\_CPARx and DMA\_CMARx).

This case is not supported by the DMA hardware, that does not guarantee that the remaining data transfers are performed correctly.

- Stop and abort a channel

If the application does not need any more the channel, this active channel can be disabled by software. The channel is stopped and aborted but the DMA\_CNDTRx register content may not correctly reflect the remaining data transfers versus the aborted source and destination buffer/register.

- Abort and restart a channel

This corresponds to the software sequence: disable an active channel, then reconfigure the channel and enable it again.

This is supported by the hardware if the following conditions are met:

- The application guarantees that, when the software is disabling the channel, a DMA data transfer is not occurring at the same time over its master port. For example, the application can first disable the peripheral in DMA mode, in order to ensure that there is no pending hardware DMA request from this peripheral.
- The software must operate separated write accesses to the same DMA\_CCRx register: First disable the channel. Second reconfigure the channel for a next block transfer including the DMA\_CCRx if a configuration change is needed. There are read-only DMA\_CCRx register fields when DMA\_CCRx.EN=1. Finally enable again the channel.

When a channel transfer error occurs, the EN bit of the DMA\_CCRx register is cleared by hardware. This EN bit can not be set again by software to re-activate the channel x, until the TEIFx bit of the DMA\_ISR register is set.

### Circular mode (in memory-to-peripheral/peripheral-to-memory transfers)

The circular mode is available to handle circular buffers and continuous data flows (such as ADC scan mode). This feature is enabled using the CIRC bit in the DMA\_CCRx register.

**Note:**

*The circular mode must not be used in memory-to-memory mode. Before enabling a channel in circular mode (CIRC = 1), the software must clear the MEM2MEM bit of the DMA\_CCRx register. When the circular mode is activated, the amount of data to transfer is*

*automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.*

*In order to stop a circular transfer, the software needs to stop the peripheral from generating DMA requests (such as quit the ADC scan mode), before disabling the DMA channel.*

*The software must explicitly program the DMA\_CNDTRx value before starting/enabling a transfer, and after having stopped a circular transfer.*

### Memory-to-memory mode

The DMA channels may operate without being triggered by a request from a peripheral. This mode is called memory-to-memory mode, and is initiated by software.

If the MEM2MEM bit in the DMA\_CCRx register is set, the channel, if enabled, initiates transfers. The transfer stops once the DMA\_CNDTRx register reaches zero.

**Note:** *The memory-to-memory mode must not be used in circular mode. Before enabling a channel in memory-to-memory mode (MEM2MEM = 1), the software must clear the CIRC bit of the DMA\_CCRx register.*

### Peripheral-to-peripheral mode

Any DMA channel can operate in peripheral-to-peripheral mode:

- when the hardware request from a peripheral is selected to trigger the DMA channel  
This peripheral is the DMA initiator and paces the data transfer from/to this peripheral to/from a register belonging to another memory-mapped peripheral (this one being not configured in DMA mode).
- when no peripheral request is selected and connected to the DMA channel  
The software configures a register-to-register transfer by setting the MEM2MEM bit of the DMA\_CCRx register.

### Programming transfer direction, assigning source/destination

The value of the DIR bit of the DMA\_CCRx register sets the direction of the transfer, and consequently, it identifies the source and the destination, regardless the source/destination type (peripheral or memory):

- **DIR = 1** defines typically a memory-to-peripheral transfer. More generally, if DIR = 1:
  - The **source** attributes are defined by the DMA\_MARx register, the MSIZE[1:0] field and MINC bit of the DMA\_CCRx register.  
Regardless of their usual naming, these 'memory' register, field and bit are used to define the source peripheral in peripheral-to-peripheral mode.
  - The **destination** attributes are defined by the DMA\_PARx register, the PSIZE[1:0] field and PINC bit of the DMA\_CCRx register.  
Regardless of their usual naming, these 'peripheral' register, field and bit are used to define the destination memory in memory-to-memory mode.
- **DIR = 0** defines typically a peripheral-to-memory transfer. More generally, if DIR = 0:
  - The **source** attributes are defined by the DMA\_PARx register, the PSIZE[1:0] field and PINC bit of the DMA\_CCRx register.  
Regardless of their usual naming, these 'peripheral' register, field and bit are used to define the source memory in memory-to-memory mode
  - The **destination** attributes are defined by the DMA\_MARx register, the MSIZE[1:0] field and MINC bit of the DMA\_CCRx register.

Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the destination peripheral in peripheral-to-peripheral mode.

### 11.4.6 DMA data width, alignment and endianness

When PSIZE[1:0] and MSIZE[1:0] are not equal, the DMA controller performs some data alignments as described in the table below.

**Table 49. Programmable data width and endian behavior (when PINC = MINC = 1)**

Source port width (MSIZE if DIR = 1, else PSIZE)	Destination port width (PSIZE if DIR = 1, else MSIZE)	Number of data items to transfer (NDT)	Source content: address / data (DMA_CMARx if DIR = 1, else DMA_CPARx)	DMA transfers	Destination content: address / data (DMA_CPARx if DIR = 1, else DMA_CMARx)
8	8	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write B0[7:0] @0x0 2: read B1[7:0] @0x1 then write B1[7:0] @0x1 3: read B2[7:0] @0x2 then write B2[7:0] @0x2 4: read B3[7:0] @0x3 then write B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8	16	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write 00B0[15:0] @0x0 2: read B1[7:0] @0x1 then write 00B1[15:0] @0x2 3: read B2[7:0] @0x2 then write 00B2[15:0] @0x4 4: read B3[7:0] @0x3 then write 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8	32	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write 000000B0[31:0] @0x0 2: read B1[7:0] @0x1 then write 000000B1[31:0] @0x4 3: read B2[7:0] @0x2 then write 000000B2[31:0] @0x8 4: read B3[7:0] @0x3 then write 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write B0[7:0] @0x0 2: read B3B2[15:0] @0x2 then write B2[7:0] @0x1 3: read B5B4[15:0] @0x4 then write B4[7:0] @0x2 4: read B7B6[15:0] @0x6 then write B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16	16	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write B1B0[15:0] @0x0 2: read B3B2[15:0] @0x2 then write B3B2[15:0] @0x2 3: read B5B4[15:0] @0x4 then write B5B4[15:0] @0x4 4: read B7B6[15:0] @0x6 then write B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16	32	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write 0000B1B0[31:0] @0x0 2: read B3B2[15:0] @0x2 then write 0000B3B2[31:0] @0x4 3: read B5B4[15:0] @0x4 then write 0000B5B4[31:0] @0x8 4: read B7B6[15:0] @0x6 then write 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32	8	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B0[7:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B4[7:0] @0x1 3: read BBBAB9B8[31:0] @0x8 then write B8[7:0] @0x2 4: read BFBEBDBC[31:0] @0xC then write BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC
32	16	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B1B0[15:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B5B4[15:0] @0x2 3: read BBBAB9B8[31:0] @0x8 then write B9B8[15:0] @0x4 4: read BFBEBDBC[31:0] @0xC then write BDBC[15:0] @0x6	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC
32	32	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B3B2B1B0[31:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B7B6B5B4[31:0] @0x4 3: read BBBAB9B8[31:0] @0x8 then write BBBAB9B8[31:0] @0x8 4: read BFBEBDBC[31:0] @0xC then write BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC

### Addressing AHB peripherals not supporting byte/half-word write transfers

When the DMA controller initiates an AHB byte or half-word write transfer, the data are duplicated on the unused lanes of the AHB master 32-bit data bus (HWDATA[31:0]).

When the AHB slave peripheral does not support byte or half-word write transfers and does not generate any error, the DMA controller writes the 32 HWDATA bits as shown in the two examples below:

- To write the half-word 0xABCD, the DMA controller sets the HWDATA bus to 0xABCDABCD with a half-word data size (HSIZE = HalfWord in AHB master bus).
- To write the byte 0xAB, the DMA controller sets the HWDATA bus to 0xABABABAB with a byte data size (HSIZE = Byte in the AHB master bus).

Assuming the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take into account the HSIZE data, any AHB byte or half-word transfer is changed into a 32-bit APB transfer as described below:

- An AHB byte write transfer of 0xB0 to one of the 0x0, 0x1, 0x2 or 0x3 addresses, is converted to an APB word write transfer of 0xB0B0B0B0 to the 0x0 address.
- An AHB half-word write transfer of 0xB1B0 to the 0x0 or 0x2 addresses, is converted to an APB word write transfer of 0xB1B0B1B0 to the 0x0 address.

#### 11.4.7 DMA error management

A DMA transfer error is generated when reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or write access, the faulty channel x is automatically disabled through a hardware clear of its EN bit in the corresponding DMA\_CCRx register.

The TEIFx bit of the DMA\_ISR register is set. An interrupt is then generated if the TEIE bit of the DMA\_CCRx register is set.

The EN bit of the DMA\_CCRx register can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA\_ISR register is cleared (by setting the CTEIFx bit of the DMA\_IFCR register).

When the software is notified with a transfer error over a channel which involves a peripheral, the software has first to stop this peripheral in DMA mode, in order to disable any pending or future DMA request. Then software may normally reconfigure both DMA and the peripheral in DMA mode for a new transfer.

## 11.5 DMA interrupts

An interrupt can be generated on a half transfer, transfer complete or transfer error for each DMA channel x. Separate interrupt enable bits are available for flexibility.

Table 50. DMA interrupt requests

Interrupt request	Interrupt event	Event flag	Interrupt enable bit
Channel x interrupt	Half transfer on channel x	HTIFx	HTIEx
	Transfer complete on channel x	TCIFx	TCIEx
	Transfer error on channel x	TEIFx	TEIEx
	Half transfer or transfer complete or transfer error on channel x	GIFx	-

## 11.6 DMA registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

The DMA registers have to be accessed by words (32-bit).

### 11.6.1 DMA interrupt status register (DMA\_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

Every status bit is cleared by hardware when the software sets the corresponding clear bit or the corresponding global clear bit CGIFx, in the DMA\_IFCR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 27 **TEIF7**: transfer error (TE) flag for channel 7

- 0: no TE event
- 1: a TE event occurred

Bit 26 **HTIF7**: half transfer (HT) flag for channel 7

- 0: no HT event
- 1: a HT event occurred

Bit 25 **TCIF7**: transfer complete (TC) flag for channel 7

- 0: no TC event
- 1: a TC event occurred

Bit 24 **GIF7**: global interrupt flag for channel 7

- 0: no TE, HT or TC event
- 1: a TE, HT or TC event occurred

- Bit 23 **TEIF6**: transfer error (TE) flag for channel 6  
0: no TE event  
1: a TE event occurred
- Bit 22 **HTIF6**: half transfer (HT) flag for channel 6  
0: no HT event  
1: a HT event occurred
- Bit 21 **TCIF6**: transfer complete (TC) flag for channel 6  
0: no TC event  
1: a TC event occurred
- Bit 20 **GIF6**: global interrupt flag for channel 6  
0: no TE, HT or TC event  
1: a TE, HT or TC event occurred
- Bit 19 **TEIF5**: transfer error (TE) flag for channel 5  
0: no TE event  
1: a TE event occurred
- Bit 18 **HTIF5**: half transfer (HT) flag for channel 5  
0: no HT event  
1: a HT event occurred
- Bit 17 **TCIF5**: transfer complete (TC) flag for channel 5  
0: no TC event  
1: a TC event occurred
- Bit 16 **GIF5**: global interrupt flag for channel 5  
0: no TE, HT or TC event  
1: a TE, HT or TC event occurred
- Bit 15 **TEIF4**: transfer error (TE) flag for channel 4  
0: no TE event  
1: a TE event occurred
- Bit 14 **HTIF4**: half transfer (HT) flag for channel 4  
0: no HT event  
1: a HT event occurred
- Bit 13 **TCIF4**: transfer complete (TC) flag for channel 4  
0: no TC event  
1: a TC event occurred
- Bit 12 **GIF4**: global interrupt flag for channel 4  
0: no TE, HT or TC event  
1: a TE, HT or TC event occurred
- Bit 11 **TEIF3**: transfer error (TE) flag for channel 3  
0: no TE event  
1: a TE event occurred
- Bit 10 **HTIF3**: half transfer (HT) flag for channel 3  
0: no HT event  
1: a HT event occurred
- Bit 9 **TCIF3**: transfer complete (TC) flag for channel 3  
0: no TC event  
1: a TC event occurred

- Bit 8 **GIF3**: global interrupt flag for channel 3  
0: no TE, HT or TC event  
1: a TE, HT or TC event occurred
- Bit 7 **TEIF2**: transfer error (TE) flag for channel 2  
0: no TE event  
1: a TE event occurred
- Bit 6 **HTIF2**: half transfer (HT) flag for channel 2  
0: no HT event  
1: a HT event occurred
- Bit 5 **TCIF2**: transfer complete (TC) flag for channel 2  
0: no TC event  
1: a TC event occurred
- Bit 4 **GIF2**: global interrupt flag for channel 2  
0: no TE, HT or TC event  
1: a TE, HT or TC event occurred
- Bit 3 **TEIF1**: transfer error (TE) flag for channel 1  
0: no TE event  
1: a TE event occurred
- Bit 2 **HTIF1**: half transfer (HT) flag for channel 1  
0: no HT event  
1: a HT event occurred
- Bit 1 **TCIF1**: transfer complete (TC) flag for channel 1  
0: no TC event  
1: a TC event occurred
- Bit 0 **GIF1**: global interrupt flag for channel 1  
0: no TE, HT or TC event  
1: a TE, HT or TC event occurred

### 11.6.2 DMA interrupt flag clear register (DMA\_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

Setting the global clear bit CGIFx of the channel x in this DMA\_IFCR register, causes the DMA hardware to clear the corresponding GIFx bit and any individual flag among TEIFx, HTIFx, TCIFx, in the DMA\_ISR register.

Setting any individual clear bit among CTEIFx, CHTIFx, CTCIFx in this DMA\_IFCR register, causes the DMA hardware to clear the corresponding individual flag and the global flag GIFx in the DMA\_ISR register, provided that none of the two other individual flags is set.

Writing 0 into any flag clear bit has no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bit 27 **CTEIF7**: transfer error flag clear for channel 7

Bit 26 **CHTIF7**: half transfer flag clear for channel 7

Bit 25 **CTCIF7**: transfer complete flag clear for channel 7

Bit 24 **CGIF7**: global interrupt flag clear for channel 7

Bit 23 **CTEIF6**: transfer error flag clear for channel 6

Bit 22 **CHTIF6**: half transfer flag clear for channel 6

Bit 21 **CTCIF6**: transfer complete flag clear for channel 6

Bit 20 **CGIF6**: global interrupt flag clear for channel 6

Bit 19 **CTEIF5**: transfer error flag clear for channel 5

Bit 18 **CHTIF5**: half transfer flag clear for channel 5

Bit 17 **CTCIF5**: transfer complete flag clear for channel 5

Bit 16 **CGIF5**: global interrupt flag clear for channel 5

Bit 15 **CTEIF4**: transfer error flag clear for channel 4

Bit 14 **CHTIF4**: half transfer flag clear for channel 4

Bit 13 **CTCIF4**: transfer complete flag clear for channel 4

Bit 12 **CGIF4**: global interrupt flag clear for channel 4

Bit 11 **CTEIF3**: transfer error flag clear for channel 3

Bit 10 **CHTIF3**: half transfer flag clear for channel 3

Bit 9 **CTCIF3**: transfer complete flag clear for channel 3

- Bit 8 **CGIF3**: global interrupt flag clear for channel 3
- Bit 7 **CTEIF2**: transfer error flag clear for channel 2
- Bit 6 **CHTIF2**: half transfer flag clear for channel 2
- Bit 5 **CTCIF2**: transfer complete flag clear for channel 2
- Bit 4 **CGIF2**: global interrupt flag clear for channel 2
- Bit 3 **CTEIF1**: transfer error flag clear for channel 1
- Bit 2 **CHTIF1**: half transfer flag clear for channel 1
- Bit 1 **CTCIF1**: transfer complete flag clear for channel 1
- Bit 0 **CGIF1**: global interrupt flag clear for channel 1

### 11.6.3 DMA channel x configuration register (DMA\_CCRx)

Address offset:  $0x08 + 0x14 * (x - 1)$ , ( $x = 1$  to 7)

Reset value: 0x0000 0000

The register fields/bits MEM2MEM, PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, and DIR are read-only when EN = 1.

The states of MEM2MEM and CIRC bits must not be both high at the same time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2 MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **MEM2MEM**: memory-to-memory mode

0: disabled

1: enabled

*Note: this bit is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is read-only when the channel is enabled (EN = 1).*

Bits 13:12 **PL[1:0]**: priority level

00: low

01: medium

10: high

11: very high

*Note: this field is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is read-only when the channel is enabled (EN = 1).*

Bits 11:10 **MSIZE[1:0]**: memory size

Defines the data size of each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: reserved

*Note: this field is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is read-only when the channel is enabled (EN = 1).*

Bits 9:8 **PSIZE[1:0]**: peripheral size

Defines the data size of each DMA transfer to the identified peripheral.

In memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: reserved

*Note: this field is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is read-only when the channel is enabled (EN = 1).*

Bit 7 **MINC**: memory increment mode

Defines the increment mode for each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

0: disabled

1: enabled

*Note: this bit is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is read-only when the channel is enabled (EN = 1).*

Bit 6 **PINC**: peripheral increment mode

Defines the increment mode for each DMA transfer to the identified peripheral.

In memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

0: disabled

1: enabled

*Note: this bit is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is read-only when the channel is enabled (EN = 1).*

Bit 5 **CIRC**: circular mode

- 0: disabled
- 1: enabled

*Note: this bit is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is not read-only when the channel is enabled (EN = 1).*

Bit 4 **DIR**: data transfer direction

This bit must be set only in memory-to-peripheral and peripheral-to-memory modes.

- 0: read from peripheral

- Source attributes are defined by PSIZE and PINC, plus the DMA\_CPARx register.  
This is still valid in a memory-to-memory mode.
- Destination attributes are defined by MSIZE and MINC, plus the DMA\_CMARx register. This is still valid in a peripheral-to-peripheral mode.

- 1: read from memory

- Destination attributes are defined by PSIZE and PINC, plus the DMA\_CPARx register. This is still valid in a memory-to-memory mode.
- Source attributes are defined by MSIZE and MINC, plus the DMA\_CMARx register.  
This is still valid in a peripheral-to-peripheral mode.

*Note: this bit is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is read-only when the channel is enabled (EN = 1).*

Bit 3 **TEIE**: transfer error interrupt enable

- 0: disabled
- 1: enabled

*Note: this bit is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is not read-only when the channel is enabled (EN = 1).*

Bit 2 **HTIE**: half transfer interrupt enable

- 0: disabled
- 1: enabled

*Note: this bit is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is not read-only when the channel is enabled (EN = 1).*

Bit 1 **TCIE**: transfer complete interrupt enable

- 0: disabled
- 1: enabled

*Note: this bit is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is not read-only when the channel is enabled (EN = 1).*

Bit 0 **EN**: channel enable

When a channel transfer error occurs, this bit is cleared by hardware. It can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA\_ISR register is cleared (by setting the CTEIFx bit of the DMA\_IFCR register).

- 0: disabled
- 1: enabled

*Note: this bit is set and cleared by software.*

#### 11.6.4 DMA channel x number of data to transfer register (DMA\_CNDTR $x$ )

Address offset: 0x0C + 0x14 \* (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **NDT[15:0]**: number of data to transfer (0 to  $2^{16} - 1$ )

This field is updated by hardware when the channel is enabled:

- It is decremented after each single DMA 'read followed by write' transfer, indicating the remaining amount of data items to transfer.
- It is kept at zero when the programmed amount of data to transfer is reached, if the channel is not in circular mode (CIRC = 0 in the DMA\_CCR $x$  register).
- It is reloaded automatically by the previously programmed value, when the transfer is complete, if the channel is in circular mode (CIRC = 1).

If this field is zero, no transfer can be served whatever the channel status (enabled or not).

*Note: this field is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is read-only when the channel is enabled (EN = 1).*

#### 11.6.5 DMA channel x peripheral address register (DMA\_CPAR $x$ )

Address offset: 0x10 + 0x14 \* (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

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

Bits 31:0 **PA[31:0]**: peripheral address

It contains the base address of the peripheral data register from/to which the data is read/written.

When PSIZE[1:0] = 01 (16 bits), bit 0 of PA[31:0] is ignored. Access is automatically aligned to a half-word address.

When PSIZE = 10 (32 bits), bits 1 and 0 of PA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this register identifies the memory destination address if DIR = 1 and the memory source address if DIR = 0.

In peripheral-to-peripheral mode, this register identifies the peripheral destination address DIR = 1 and the peripheral source address if DIR = 0.

*Note: this register is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is not read-only when the channel is enabled (EN = 1).*

## 11.6.6 DMA channel x memory address register (DMA\_CMARx)

Address offset: 0x14 + 0x14 \* (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

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

Bits 31:0 **MA[31:0]**: peripheral address

It contains the base address of the memory from/to which the data is read/written.

When MSIZE[1:0] = 01 (16 bits), bit 0 of MA[31:0] is ignored. Access is automatically aligned to a half-word address.

When MSIZE = 10 (32 bits), bits 1 and 0 of MA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this register identifies the memory source address if DIR = 1 and the memory destination address if DIR = 0.

In peripheral-to-peripheral mode, this register identifies the peripheral source address DIR = 1 and the peripheral destination address if DIR = 0.

*Note: this register is set and cleared by software.*

*It must not be written when the channel is enabled (EN = 1).*

*It is not read-only when the channel is enabled (EN = 1).*

## 11.6.7 DMA register map

The table below gives the DMA register map and reset values.

Table 51. DMA register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMA_ISR	Res	Res	Res	Res	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 51. 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
0x004	DMA_IFCR	Res.																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DMA_CCR1	Res.																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	DMA_CNDTR1	Res.																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	DMA_CPAR1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_CMAR1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	DMA_CCR2	Res.																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DMA_CNDTR2	Res.																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DMA_CPAR2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DMA_CMAR2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	DMA_CCR3	Res.																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	DMA_CNDTR3	Res.																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DMA_CPAR3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	DMA_CMAR3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	DMA_CCR4	Res.																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	DMA_CNDTR4	Res.																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	DMA_CPAR4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	DMA_CMAR4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved	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 51. DMA register map and reset values (continued)**

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

## 12 DMA request multiplexer (DMAMUX)

### 12.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until it is served by the DMA controller that generates a DMA acknowledge signal, and the corresponding DMA request signal is deasserted.

In this document, the set of control signals required for the DMA request/acknowledge protocol is not explicitly shown or described, and it is referred to as DMA request line.

The DMAMUX request multiplexer enables routing a DMA request line between the peripherals and the DMA controllers of the product. The routing function is ensured by a programmable multi-channel DMA request line multiplexer. Each channel selects a unique DMA request line, unconditionally or synchronously with events from its DMAMUX synchronization inputs. The DMAMUX may also be used as a DMA request generator from programmable events on its input trigger signals.

The number of DMAMUX instances and their main characteristics are specified in [Section 12.3.1](#).

The assignment of DMAMUX request multiplexer inputs to the DMA request lines from peripherals and to the DMAMUX request generator outputs, the assignment of DMAMUX request multiplexer outputs to DMA controller channels, and the assignment of DMAMUX synchronizations and trigger inputs to internal and external signals depend on the product implementation, and are detailed in [Section 12.3.2](#).

## 12.2 DMAMUX main features

- 14-channel programmable DMA request line multiplexer output
- 4-channel DMA request generator
- 20 trigger inputs to DMA request generator
- 20 synchronization inputs
- Per DMA request generator channel:
  - DMA request trigger input selector
  - DMA request counter
  - Event overrun flag for selected DMA request trigger input
- Per DMA request line multiplexer channel output:
  - 36 input DMA request lines from peripherals
  - One DMA request line output
  - Synchronization input selector
  - DMA request counter
  - Event overrun flag for selected synchronization input
  - One event output, for DMA request chaining

## 12.3 DMAMUX implementation

Table 52. DMAMUX implementation

Feature	STM32WB55xx	STM32WB35xx
DMA request SPI2	X	-

### 12.3.1 DMAMUX instantiation

DMAMUX is instantiated with the hardware configuration parameters listed in the following table.

Table 53. DMAMUX instantiation

Feature	DMAMUX
Number of DMAMUX output request channels	14
Number of DMAMUX request generator channels	4
Number of DMAMUX request trigger inputs	20
Number of DMAMUX synchronization inputs	20
Number of DMAMUX peripheral request inputs	36

### 12.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.

DMAMUX is used with DMA1 and DMA2:

- DMAMUX channels 0 to 6 are connected to DMA1 channels 1 to 7
- DMAMUX channels 7 to 13 are connected to DMA2 channels 1 to 7

**Table 54. DMAMUX: assignment of multiplexer inputs to resources**

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
1	dmamux_req_gen0	22	TIM1_CH2	43	Reserved
2	dmamux_req_gen1	23	TIM1_CH3	44	Reserved
3	dmamux_req_gen2	24	TIM1_CH4	45	Reserved
4	dmamux_req_gen3	25	TIM1_UP	46	Reserved
5	ADC1	26	TIM1_TRIG	47	Reserved
6	SPI1_RX	27	TIM1_COM	48	Reserved
7	SPI1_TX	28	TIM2_CH1	49	Reserved
8	SPI2_RX	29	TIM2_CH2	50	Reserved
9	SPI2_TX	30	TIM2_CH3	51	Reserved
10	I2C1_RX	31	TIM2_CH4	52	Reserved
11	I2C1_TX	32	TIM2_UP	53	Reserved
12	I2C3_RX	33	TIM16_CH1	54	Reserved
13	I2C3_TX	34	TIM16_UP	55	Reserved
14	USART1_RX	35	TIM17_CH1	56	Reserved
15	USART1_TX	36	TIM17_UP	57	Reserved
16	LPUART1_RX	37	AES1_IN	58	Reserved
17	LPUART1_TX	38	AES1_OUT	59	Reserved
18	SAI1_A	39	AES2_IN	60	Reserved
19	SAI1_B	40	AES2_OUT	61	Reserved
20	QUADSPI	41	Reserved	62	Reserved
21	TIM1_CH1	42	Reserved	63	Reserved

**Table 55. DMAMUX: assignment of trigger inputs to resources**

Trigger input	Resource	Trigger input	Resource
0	EXTI LINE0	16	dmamux_evt0
1	EXTI LINE1	17	dmamux_evt1
2	EXTI LINE2	18	LPTIM1_OUT
3	EXTI LINE3	19	LPTIM2_OUT
4	EXTI LINE4	20	Reserved
5	EXTI LINE5	21	Reserved
6	EXTI LINE6	22	Reserved
7	EXTI LINE7	23	Reserved

**Table 55. DMAMUX: assignment of trigger inputs to resources (continued)**

Trigger input	Resource	Trigger input	Resource
8	EXTI LINE8	24	Reserved
9	EXTI LINE9	25	Reserved
10	EXTI LINE10	26	Reserved
11	EXTI LINE11	27	Reserved
12	EXTI LINE12	28	Reserved
13	EXTI LINE13	29	Reserved
14	EXTI LINE14	30	Reserved
15	EXTI LINE15	31	Reserved

**Table 56. DMAMUX: assignment of synchronization inputs to resources**

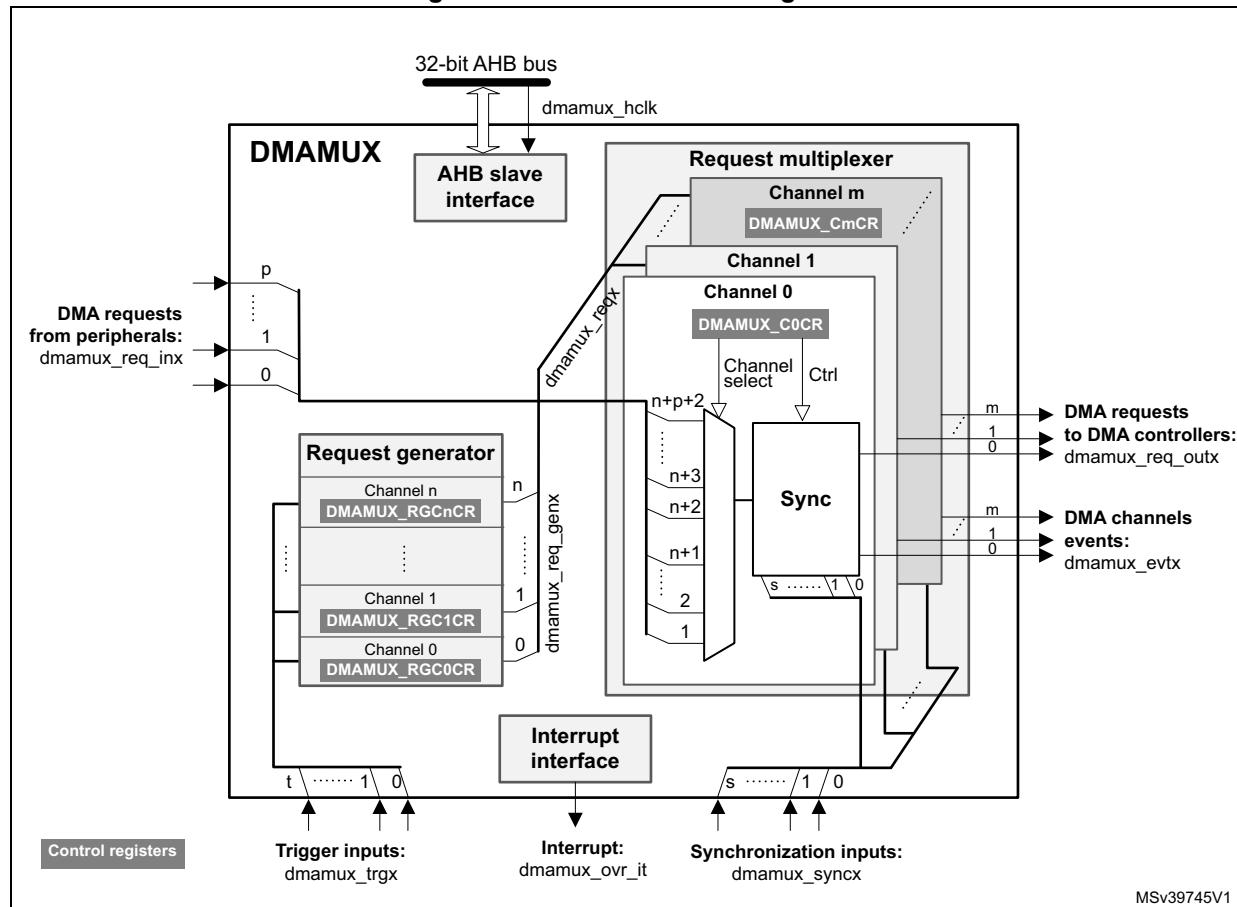
Sync. input	Resource	Sync. input	Resource
0	EXTI LINE0	16	dmamux_evt0
1	EXTI LINE1	17	dmamux_evt1
2	EXTI LINE2	18	LPTIM1_OUT
3	EXTI LINE3	19	LPTIM2_OUT
4	EXTI LINE4	20	Reserved
5	EXTI LINE5	21	Reserved
6	EXTI LINE6	22	Reserved
7	EXTI LINE7	23	Reserved
8	EXTI LINE8	24	Reserved
9	EXTI LINE9	25	Reserved
10	EXTI LINE10	26	Reserved
11	EXTI LINE11	27	Reserved
12	EXTI LINE12	28	Reserved
13	EXTI LINE13	29	Reserved
14	EXTI LINE14	30	Reserved
15	EXTI LINE15	31	Reserved

## 12.4 DMAMUX functional description

### 12.4.1 DMAMUX block diagram

Figure 26 shows the DMAMUX block diagram.

Figure 26. DMAMUX block diagram



MSv39745V1

DMAMUX features two main sub-blocks: the request line multiplexer and the request line generator.

The implementation assigns:

- DMAMUX request multiplexer sub-block inputs (`dmamux_reqx`) from peripherals (`dmamux_req_inx`) and from channels of the DMAMUX request generator sub-block (`dmamux_req_genx`)
- DMAMUX request outputs to channels of DMA controllers (`dmamux_req_outx`)
- Internal or external signals to DMA request trigger inputs (`dmamux_trgx`)
- Internal or external signals to synchronization inputs (`dmamux_syncx`)

### 12.4.2 DMAMUX signals

*Table 57* lists the DMAMUX signals.

**Table 57. DMAMUX signals**

Signal name	Description
dmamux_hclk	DMAMUX AHB clock
dmamux_req_inx	DMAMUX DMA request line inputs from peripherals
dmamux_trgx	DMAMUX DMA request triggers inputs (to request generator sub-block)
dmamux_req_genx	DMAMUX request generator sub-block channels outputs
dmamux_reqx	DMAMUX request multiplexer sub-block inputs (from peripheral requests and request generator channels)
dmamux_syncx	DMAMUX synchronization inputs (to request multiplexer sub-block)
dmamux_req_outx	DMAMUX requests outputs (to DMA controllers)
dmamux_evtx	DMAMUX events outputs
dmamux_ovr_it	DMAMUX overrun interrupts

### 12.4.3 DMAMUX channels

A DMAMUX channel is a DMAMUX request multiplexer channel that may include, depending on the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to one single channel of DMA controller(s).

#### Channel configuration procedure

Follow the sequence below to configure both a DMAMUX x channel and the related DMA channel y:

1. Set and configure completely the DMA channel y, except enabling the channel y.
2. Set and configure completely the related DMAMUX y channel.
3. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

### 12.4.4 DMAMUX request line multiplexer

The DMAMUX request multiplexer with its multiple channels ensures the actual routing of DMA request/acknowledge control signals, named DMA request lines.

Each DMA request line is connected in parallel to all the channels of the DMAMUX request line multiplexer.

A DMA request is sourced either from the peripherals or from the DMAMUX request generator.

The DMAMUX request line multiplexer channel x selects the DMA request line number as configured by the DMAREQ\_ID field in the DMAMUX\_CxCR register.

*Note:* *The null value in the field DMAREQ\_ID corresponds to no DMA request line selected.*

**Caution:** A same non-null DMAREQ\_ID must not be programmed to different x and y DMAMUX request multiplexer channels (via DMAMUX\_CxCR and DMAMUX\_CyCR), except if application guarantees that the two connected DMA channels are not simultaneously active.

On top of the DMA request selection, the synchronization mode and/or the event generation may be configured and enabled, if required.

### Synchronization mode and channel event generation

Each DMAMUX request line multiplexer channel x can be individually synchronized by setting the synchronization enable (SE) bit in the DMAMUX\_CxCR register.

DMAMUX has multiple synchronization inputs. The synchronization inputs are connected in parallel to all the channels of the request multiplexer.

The synchronization input is selected via the SYNC\_ID field in the DMAMUX\_CxCR register of a given channel x.

When a channel is in this synchronization mode, the selected input DMA request line is propagated to the multiplexer channel output, once is detected a programmable rising/falling edge on the selected input synchronization signal, via the SPOL[1:0] field of the DMAMUX\_CxCR register.

Additionally, there is a programmable DMA request counter, internally to the DMAMUX request multiplexer, which may be used for the channel request output generation and also possibly for an event generation. An event generation on the channel x output is enabled through the EGE bit (event generation enable) of the DMAMUX\_CxCR register.

As shown in [Figure 28](#), upon the detected edge of the synchronization input, the pending selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

**Note:** *If a synchronization event occurs while there is no pending selected input DMA request line, it is discarded. The following asserted input request lines is not connected to the DMAMUX multiplexer channel output until a synchronization event occurs again.*

From this point on, each time the connected DMAMUX request is served by the DMA controller (a served request is deasserted), the DMAMUX request counter is decremented. At its underrun, the DMA request counter is automatically loaded with the value in NBREQ field of the DMAMUX\_CxCR register and the input DMA request line is disconnected from the multiplexer channel x output.

Thus, the number of DMA requests transferred to the multiplexer channel x output following a detected synchronization event, is equal to the value in NBREQ field, plus one.

**Note:** *The NBREQ field value shall only be written by software when both synchronization enable bit SE and event generation enable EGE bit of the corresponding multiplexer channel x are disabled.*

Figure 27. Synchronization mode of the DMAMUX request line multiplexer channel

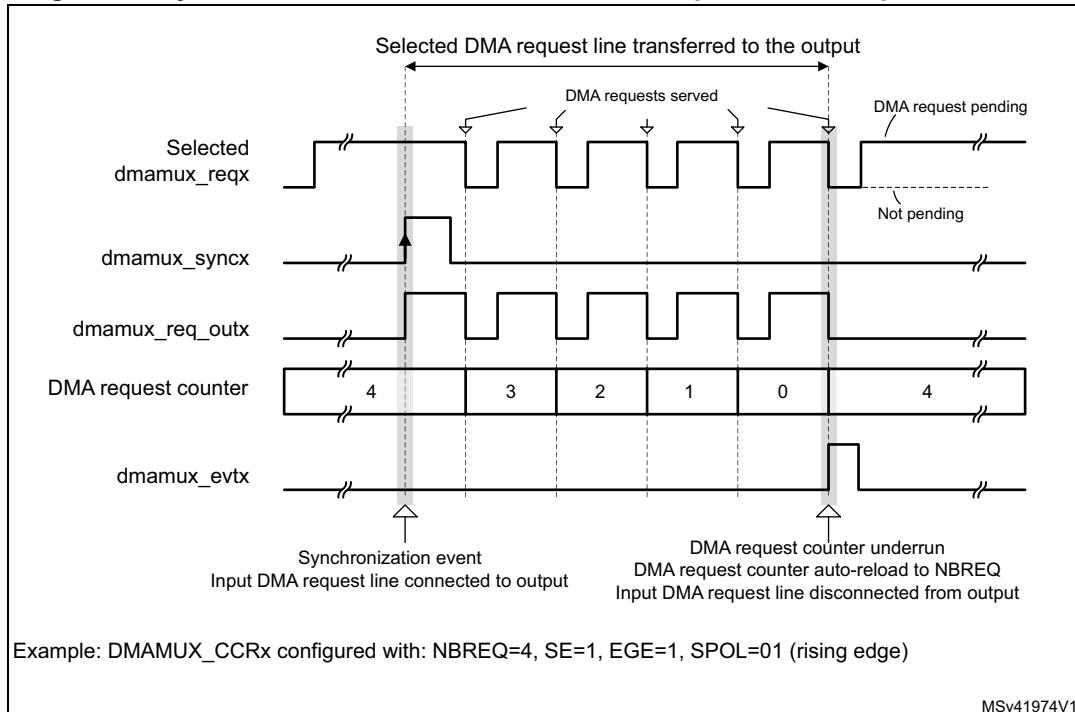
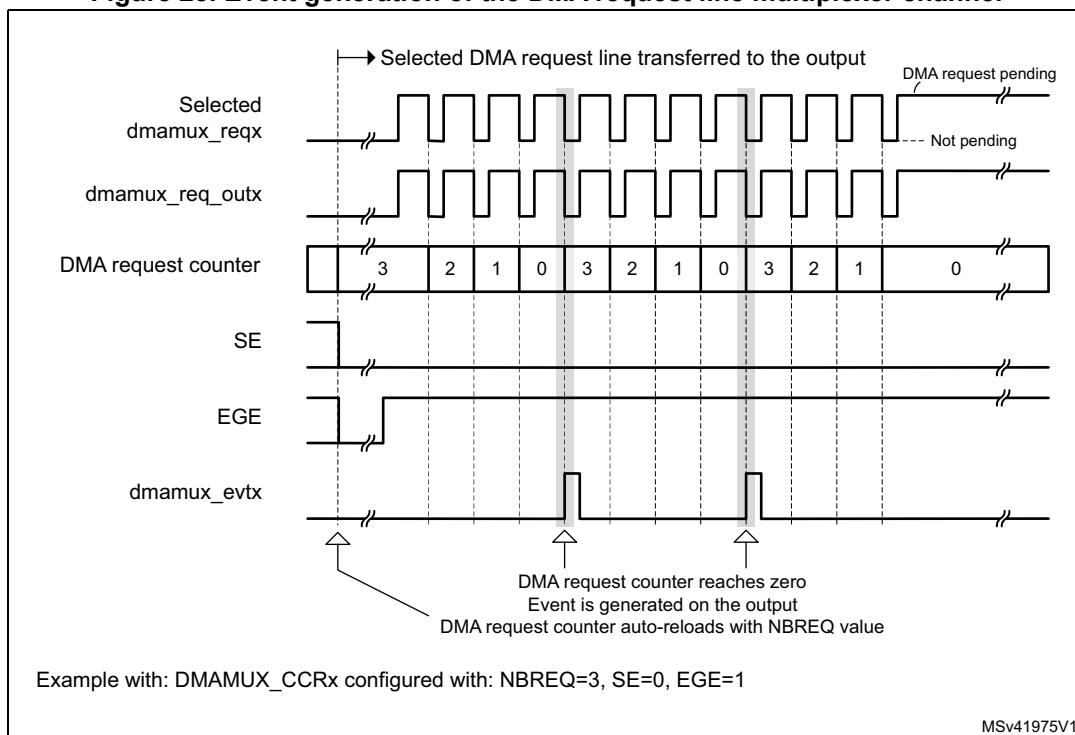


Figure 28. Event generation of the DMA request line multiplexer channel



If EGE is enabled, the multiplexer channel generates a channel event, as a pulse of one AHB clock cycle, when its DMA request counter is automatically reloaded with the value of the programmed NBREQ field, as shown in [Figure 27](#) and [Figure 28](#).

- Note:** *If EGE is enabled and NBREQ = 0, an event is generated after each served DMA request.*
- Note:** *A synchronization event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.*
- Upon writing into DMAMUX\_CxCR register, the synchronization events are masked during three AHB clock cycles.*

### Synchronization overrun and interrupt

If a new synchronization event occurs before the request counter underrun (the internal request counter programmed via the NBREQ field of the DMAMUX\_CxCR register), the synchronization overrun flag bit SOFx is set in the DMAMUX\_CSR status register.

- Note:** *The request multiplexer channel x synchronization must be disabled (DMAMUX\_CxCR.SE = 0) at the completion of the use of the related channel of the DMA controller. Else, upon a new detected synchronization event, there is a synchronization overrun due to the absence of a DMA acknowledge (that is, no served request) received from the DMA controller.*

The overrun flag SOFx is reset by setting the associated clear synchronization overrun flag bit CSOFx in the DMAMUX\_CFR register.

Setting the synchronization overrun flag generates an interrupt if the synchronization overrun interrupt enable bit SOIE is set in the DMAMUX\_CxCR register.

## 12.4.5 DMAMUX request generator

The DMAMUX request generator produces DMA requests following trigger events on its DMA request trigger inputs.

The DMAMUX request generator has multiple channels. DMA request trigger inputs are connected in parallel to all channels.

The outputs of DMAMUX request generator channels are inputs to the DMAMUX request line multiplexer.

Each DMAMUX request generator channel x has an enable bit GE (generator enable) in the corresponding DMAMUX\_RGxCR register.

The DMA request trigger input for the DMAMUX request generator channel x is selected through the SIG\_ID (trigger signal ID) field in the corresponding DMAMUX\_RGxCR register.

Trigger events on a DMA request trigger input can be rising edge, falling edge or either edge. The active edge is selected through the GPOL (generator polarity) field in the corresponding DMAMUX\_RGxCR register.

Upon the trigger event, the corresponding generator channel starts generating DMA requests on its output. Each time the DMAMUX generated request is served by the connected DMA controller (a served request is deasserted), a built-in (inside the DMAMUX request generator) DMA request counter is decremented. At its underrun, the request generator channel stops generating DMA requests and the DMA request counter is automatically reloaded to its programmed value upon the next trigger event.

Thus, the number of DMA requests generated after the trigger event is GNBREQ + 1.

**Note:** The GNBREQ field value must be written by software only when the enable GE bit of the corresponding generator channel x is disabled.

*There is no hardware write protection.*

*A trigger event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.*

*Upon writing into DMAMUX\_RGxCR register, the trigger events are masked during three AHB clock cycles.*

### Trigger overrun and interrupt

If a new DMA request trigger event occurs before the DMAMUX request generator counter underrun (the internal counter programmed via the GNBREQ field of the DMAMUX\_RGxCR register), and if the request generator channel x was enabled via GE, then the request trigger event overrun flag bit OFx is asserted by the hardware in the status DMAMUX\_RGSR register.

**Note:** The request generator channel x must be disabled (DMAMUX\_RGxCR.GE = 0) at the completion of the usage of the related channel of the DMA controller. Else, upon a new detected trigger event, there is a trigger overrun due to the absence of an acknowledge (that is, no served request) received from the DMA.

The overrun flag OFx is reset by setting the associated clear overrun flag bit COFx in the DMAMUX\_RGCFR register.

Setting the DMAMUX request trigger overrun flag generates an interrupt if the DMA request trigger event overrun interrupt enable bit OIE is set in the DMAMUX\_RGxCR register.

## 12.5 DMAMUX interrupts

An interrupt can be generated upon:

- a synchronization event overrun in each DMA request line multiplexer channel
- a trigger event overrun in each DMA request generator channel

For each case, per-channel individual interrupt enable, status and clear flag register bits are available.

**Table 58. DMAMUX interrupts**

Interrupt signal	Interrupt event	Event flag	Clear bit	Enable bit
dmamuxovr_it	Synchronization event overrun on channel x of the DMAMUX request line multiplexer	SOFx	CSOFx	SOIE
	Trigger event overrun on channel x of the DMAMUX request generator	OFx	COFx	OIE

## 12.6 DMAMUX registers

Refer to the table containing register boundary addresses for the DMAMUX base address.

DMAMUX registers may be accessed per (8-bit) byte, (16-bit) half-word, or (32-bit) word. The address must be aligned with the data size.

### 12.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX\_CxCR)

Address offset: 0x000 + 0x04 \* x (x = 0 to 13)

Reset value: 0x0000 0000

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SYNC\_ID[4:0]**: Synchronization identification

Selects the synchronization input (see [Table 56: DMAMUX: assignment of synchronization inputs to resources](#)).

Bits 23:19 **NBREQ[4:0]**: Number of DMA requests minus 1 to forward

Defines the number of DMA requests to forward to the DMA controller after a synchronization event, and/or the number of DMA requests before an output event is generated.

This field shall only be written when both SE and EGE bits are low.

Bits 18:17 **SPOL[1:0]**: Synchronization polarity

Defines the edge polarity of the selected synchronization input:

00: No event, i.e. no synchronization nor detection.

01: Rising edge

10: Falling edge

11: Rising and falling edges

Bit 16 **SE**: Synchronization enable

0: Synchronization disabled

1: Synchronization enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **EGE**: Event generation enable

0: Event generation disabled

1: Event generation enabled

Bit 8 **SOIE**: Synchronization overrun interrupt enable

0: Interrupt disabled

1: Interrupt enabled

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

Bits 5:0 **DMAREQ\_ID[5:0]**: DMA request identification

Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

## 12.6.2 DMAMUX request line multiplexer interrupt channel status register (DMAMUX\_CSR)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SOF13	SOF12	SOF11	SOF10	SOF9	SOF8	SOF7	SOF6	SOF5	SOF4	SOF3	SOF2	SOF1	SOF0
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 13:0 **SOF[13:0]**: Synchronization overrun event flag

The flag is set when a synchronization event occurs on a DMA request line multiplexer channel x, while the DMA request counter value is lower than NBREQ.

The flag is cleared by writing 1 to the corresponding CSOFx bit in DMAMUX\_CFR register.

## 12.6.3 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX\_CFR)

Address offset: 0x084

Reset value: 0x0000 0000

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

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

Bits 13:0 **CSOF[13:0]**: Clear synchronization overrun event flag

Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX\_CSR register.

### 12.6.4 DMAMUX request generator channel x configuration register (DMAMUX\_RGxCR)

Address offset: 0x100 + 0x04 \* x (x = 0 to 3)

Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:19 **GNBREQ[4:0]**: Number of DMA requests to be generated (minus 1)

Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ +1.

*Note: This field must be written only when GE bit is disabled.*

Bits 18:17 **GPOL[1:0]**: DMA request generator trigger polarity

Defines the edge polarity of the selected trigger input

00: No event, i.e. no trigger detection nor generation.

01: Rising edge

10: Falling edge

11: Rising and falling edges

Bit 16 **GE**: DMA request generator channel x enable

0: DMA request generator channel x disabled

1: DMA request generator channel x enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **OIE**: Trigger overrun interrupt enable

0: Interrupt on a trigger overrun event occurrence is disabled

1: Interrupt on a trigger overrun event occurrence is enabled

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

Bits 4:0 **SIG\_ID[4:0]**: Signal identification

Selects the DMA request trigger input used for the channel x of the DMA request generator

### 12.6.5 DMAMUX request generator interrupt status register (DMAMUX\_RGSR)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OF3	OF2	OF1	OF0											
												r	r	r	r

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

Bits 3:0 **OF[3:0]**: Trigger overrun event flag

The flag is set when a new trigger event occurs on DMA request generator channel x, before the request counter underrun (the internal request counter programmed via the GNBREQ field of the DMAMUX\_RGxCR register).

The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX\_RGCFR register.

### 12.6.6 DMAMUX request generator interrupt clear flag register (DMAMUX\_RGCFR)

Address offset: 0x144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	COF3	COF2	COF1	COF0											
												w	w	w	w

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

Bits 3:0 **COF[3:0]**: Clear trigger overrun event flag

Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX\_RGSR register.

## 12.6.7 DMAMUX register map

The following table summarizes the DMAMUX registers and reset values. Refer to the register boundary address table for the DMAMUX register base address.

**Table 59. DMAMUX register map and reset values**

Table 59. DMAMUX register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x088 - 0x0FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
0x100	DMAMUX_RG0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0 0 0 0 0 0																																
0x104	DMAMUX_RG1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0 0 0 0 0 0																																
0x108	DMAMUX_RG2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0 0 0 0 0 0																																
0x10C	DMAMUX_RG3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0 0 0 0 0 0																																
0x110 - 0x13C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
0x140	DMAMUX_RGSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x144	DMAMUX_RGCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x148 - 0x3FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 13 Nested vectored interrupt controller (NVIC)

### 13.1 NVIC main features

The CPU1 NVIC features:

- 63 maskable interrupt channels (not including the sixteen Cortex<sup>®</sup>-M4 with FPU interrupt lines)
- 16 programmable priority levels (four bits of interrupt priority are used)
- Low-latency exception interrupt handling
- Power management control
- Implementation of System control registers

The CPU2 NVIC features:

- 32 maskable interrupt channels (not including the sixteen Cortex<sup>®</sup>-M0+ interrupt lines)
- Four programmable priority levels (two bits of interrupt priority are used)
- Low-latency exception interrupt handling
- Power management control

The NVICs and the processor cores interfaces are closely coupled, resulting in low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming refer to *STM32 Cortex<sup>®</sup>-M4 MCUs and MPUs programming manual* (PM0214) for Cortex<sup>®</sup>-M4, and *Cortex<sup>®</sup>-M0+ programming manual for STM32L0, STM32G0, STM32WL and STM32WB Series* (PM0223) for Cortex<sup>®</sup>-M0+.

### 13.2 NVIC implementation

Table 60. NVIC implementation

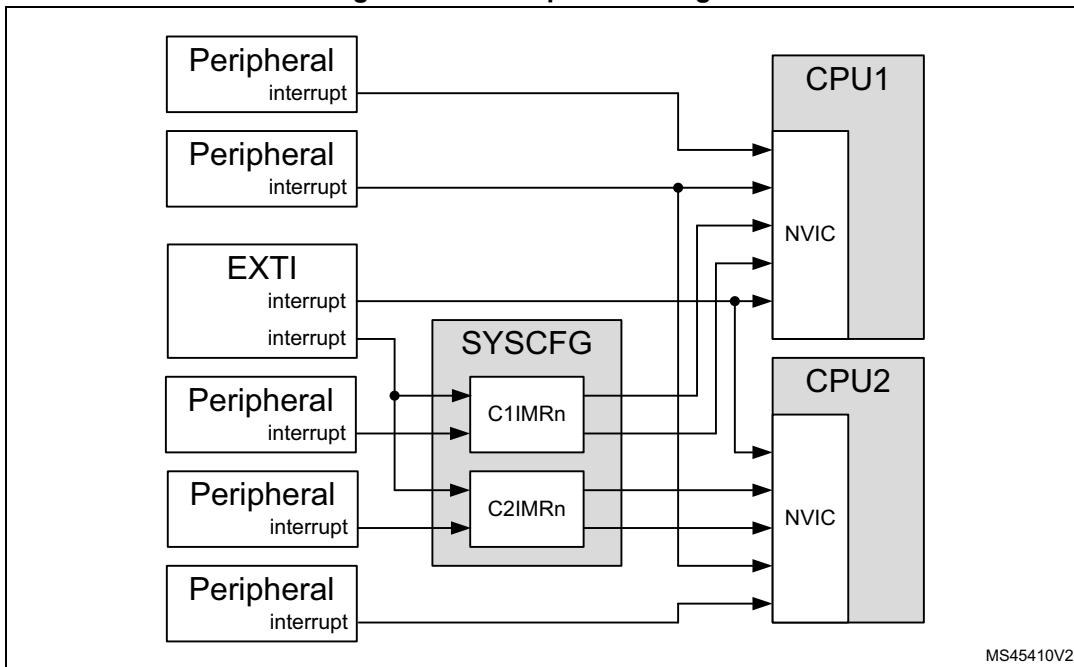
Feature	STM32WB55xx	STM32WB35xx
SPI2 interrupt	X	-
TSC interrupt	X	-
LCD interrupt	X	-
LCD wakeup	X	-

### 13.3 Interrupt block diagram

The different peripheral interrupts are connected in different ways, depending on the sharing between the two CPUs. To prevent a peripheral or EXTI interrupt to trigger both CPUs, they can be masked either in the NVIC, or, for the NVIC vector sharing multiple peripheral interrupts, by a pre-mask in the SYSCFG registers, see [Section 10: System configuration controller \(SYSCFG\)](#).

The interrupt block diagram is shown in [Figure 29](#).

Figure 29. Interrupt block diagram



## 13.4 Interrupt and exception vectors

Each of the CPU1 and CPU2 has its own vector table, see, respectively, [Table 61](#) and [Table 62](#), where shaded cells indicate the processor exceptions.

Table 61. CPU1 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 HSE CSS, Flash ECC, and SRAM2 parity	0x0000 0008
-	-1	Fixed	HardFault	All classes of fault	0x0000 000C
-	0	Settable	MemManager	Memory manager	0x0000 0010
-	1	Settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-	2	Settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C 0x0000 0028
-	3	Settable	SVCall	System service call via SWI instruction	0x0000 002C
-	4	Settable	Debug	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 early wakeup	0x0000 0040
1	8	Settable	PVD, PVM1, PVM3	PVD through EXTI[16] (C1IMR2[20]) PVM1 through EXTI[31] (C1IMR2[16]) PVM3 through EXTI[33] (C1IMR2[18])	0x0000 0044
2	9	Settable	TAMP, RTC_STAMP, LSE_CSS	Tamper,TimeStamp, LSECSS interrupt through EXTI[18]	0x0000 0048
3	10	Settable	RTC_WKUP	RTC wakeup interrupt through EXTI[19]	0x0000 004C
4	11	Settable	Flash	Flash memory global interrupt and Flash memory ECC single error interrupt	0x0000 0050
5	12	Settable	RCC	RCC global interrupt	0x0000 0054
6	13	Settable	EXTI0	EXTI line 0 interrupt through EXTI[0]	0x0000 0058
7	14	Settable	EXTI1	EXTI line 1 interrupt through EXTI[1]	0x0000 005C
8	15	Settable	EXTI2	EXTI line 2 interrupt through EXTI[2]	0x0000 0060
9	16	Settable	EXTI3	EXTI line 3 interrupt through EXTI[3]	0x0000 0064
10	17	Settable	EXTI4	EXTI line 4 interrupt through EXTI[4]	0x0000 0068
11	18	Settable	DMA1_CH1	DMA1 channel 1 interrupt	0x0000 006C
12	19	Settable	DMA1_CH2	DMA1 channel 2 interrupt	0x0000 0070
13	20	Settable	DMA1_CH3	DMA1 channel 3 interrupt	0x0000 0074
14	21	Settable	DMA1_CH4	DMA1 channel 4 interrupt	0x0000 0078
15	22	Settable	DMA1_CH5	DMA1 channel 5 interrupt	0x0000 007C

Table 61. CPU1 vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
16	23	Settable	DMA1_CH6	DMA1 channel 6 interrupt	0x0000 0080
17	24	Settable	DMA1_CH7	DMA1 channel 7 interrupt	0x0000 0084
18	25	Settable	ADC1	ADC1 global interrupt	0x0000 0088
19	26	Settable	USB_HP	USB high priority interrupt	0x0000 008C
20	27	Settable	USB_LP	USB low priority interrupt (including USB wakeup)	0x0000 0090
21	28	Settable	C2SEV PWR_C2H	CPU2 SEV through EXTI[40] PWR CPU2 HOLD wakeup interrupt	0x0000 0094
22	29	Settable	COMP	COMP2 and COMP1 interrupt through EXTI[21:20]	0x0000 0098
23	30	Settable	EXTI[9:5]	EXTI line [9:5] interrupt through EXTI[9:5] (C1IMR1[25:21])	0x0000 009C
24	31	Settable	TIM1_BRK	Timer 1 break interrupt	0x0000 00A0
25	32	Settable	TIM1_UP TIM16	Timer 1 update (C1IMR1[13]) Timer 16 global interrupt (C1IMR1[14])	0x0000 00A4
26	33	Settable	TIM1_TRG_COM TIM17	Timer 1 trigger and communication (C1IMR1[13]) Timer 17 global interrupt (C1IMR1[15])	0x0000 00A8
27	34	Settable	TIM1_CC	Timer 1 capture compare interrupt	0x0000 00AC
28	35	Settable	TIM2	Timer 2 global interrupt	0x0000 00B0
29	36	Settable	PKA	Private key accelerator interrupt	0x0000 00B4
30	37	Settable	I2C1_EV	I2C1 event interrupt	0x0000 00B8
31	38	Settable	I2C1_ER	I2C1 error interrupt	0x0000 00BC
32	39	Settable	I2C3_EV	I2C3 event interrupt	0x0000 00C0
33	40	Settable	I2C3_ER	I2C3 error interrupt	0x0000 00C4
34	41	Settable	SPI1	SPI 1 global interrupt	0x0000 00C8
35	42	Settable	SPI2	SPI 2 global interrupt	0x0000 00CC
36	43	Settable	USART1	USART1 global interrupt	0x0000 00D0
37	44	Settable	LPUART1	LPUART1 global interrupt	0x0000 00D4
38	45	Settable	SAI1	SAI1 A and B global interrupt	0x0000 00D8
39	46	Settable	TSC	TSC global interrupt	0x0000 00DC
40	47	Settable	EXTI[15:10]	EXTI line [15:10] interrupt through EXTI[15:10] (C1IMR1[31:26])	0x0000 00E0
41	48	Settable	RTC_ALARM	RTC alarms (A and B) interrupt through EXTI[17]	0x0000 00E4
42	49	Settable	CRS_IT	CRS interrupt	0x0000 00E8

Table 61. CPU1 vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
43	50	Settable	PWR_SOTF PWR_BLEACT PWR_802ACT PWR_RFPHASE	PWR switching on the fly interrupt PWR end of BLE activity interrupt PWR end of 802.15.4 activity interrupt PWR end of critical radio phase interrupt	0x0000 00EC
44	51	Settable	IPCC_C1_RX_IT	IPCC CPU1 RX occupied interrupt	0x0000 00F0
45	52	Settable	IPCC_C1_TX_IT	IPCC CPU1 TX free interrupt	0x0000 00F4
46	53	Settable	HSEM	Semaphore interrupt 0 to CPU1	0x0000 00F8
47	54	Settable	LPTIM1	LPTimer 1 global interrupt	0x0000 00FC
48	55	Settable	LPTIM2	LPTimer 2 global interrupt	0x0000 0100
49	56	Settable	LCD	LCD global interrupt	0x0000 0104
50	57	Settable	QUADSPI	QUADSPI global interrupt	0x0000 0108
51	58	Settable	AES1	AES1 global interrupt	0x0000 010C
52	59	Settable	AES2	AES2 global interrupt	0x0000 0110
53	60	Settable	True RNG	True random number generator interrupt	0x0000 0114
54	61	Settable	FPU	Floating point unit interrupt	0x0000 0118
55	62	Settable	DMA2_CH1	DMA2_channel 1 interrupt	0x0000 011C
56	63	Settable	DMA2_CH2	DMA2_channel 2 interrupt	0x0000 0120
57	64	Settable	DMA2_CH3	DMA2_channel 3 interrupt	0x0000 0124
58	65	Settable	DMA2_CH4	DMA2_channel 4 interrupt	0x0000 0128
59	66	Settable	DMA2_CH5	DMA2_channel 5 interrupt	0x0000 012C
60	67	Settable	DMA2_CH6	DMA2_channel 6 interrupt	0x0000 0130
61	68	Settable	DMA2_CH7	DMA2_channel 7 interrupt	0x0000 0134
62	69	Settable	DMAMUX1_OVR	DMAMUX1 overrun interrupt	0x0000 0138

Table 62. CPU2 vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	Fixed	Reset	Reset	0x0000 0004
-14	-2	Fixed	NMI	Non maskable interrupt HSE CSS, Flash ECC, and SRAM2 parity	0x0000 0008
-13	-1	Fixed	HardFault	All classes of fault	0x0000 000C
-	-	-	-	Reserved	0x0000 0010 0x0000 0028
-5	0	Settable	SVCall	System service call via SWI instruction	0x0000 002C
-	-	-	-	Reserved	0x0000 0030 0x0000 0034
-2	1	Settable	PendSV	Pendable request for system service	0x0000 0038
-1	2	Settable	Systick	System tick timer	0x0000 003C
0	3	Settable	-	Reserved	0x0000 0040
1	4	Settable	PVD, PVM1, PVM3	PVD through EXTI[16] (C2IMR2[20]) PVM1 through EXTI[31] (C2IMR2[16]) PVM3 through EXTI[33] (C2IMR2[18])	0x0000 0044
2	5	Settable	RTC_WKUP, TAMP, RTC_STAMP LSE_CSS, RTC_ALARM	RTC wakeup interrupt through EXTI[19] (C2IMR1[3]) Tamper, TimeStamp LSECSS interrupt through EXTI[18] (C2IMR1[0]) RTC alarms (A and B) interrupt through EXTI[17] (C2IMR1[4])	0x0000 0048
3	6	Settable	USB_HP, USB_LP CRS_IT	USB high priority interrupt, USB low priority interrupt (including USB wakeup) CRS interrupt	0x0000 004C
4	7	Settable	RCC FLASH C1SEV	RCC global interrupt (C2IMR1[5]) Flash memory global interrupt and Flash memory ECC single error interrupt (C2IMR1[6]) CPU1 SEV through EXTI[41]	0x0000 0050
5	8	Settable	EXTI[1:0]	EXTI line 1:0 interrupt through EXTI[1:0] (C2IMR1[17:16])	0x0000 0054
6	9	Settable	EXTI[3:2]	EXTI line 3:2 interrupt through EXTI[3:2] (C2IMR1[19:18])	0x0000 0058
7	10	Settable	EXTI[15:4]	EXTI line 15:4 interrupt through EXTI[15:4] (C2IMR1[31:20])	0x0000 005C
8	11	Settable	TSC 802_IT0	TSC global interrupt (C2IMR2[21]) 802.15.4 interrupt 0	0x0000 0060
9	12	Settable	DMA1_CH[3:1]	DMA1 channel 3:1 interrupt (C2IMR2[2:0])	0x0000 0064
10	13	Settable	DMA1_CH[7:4]	DMA1 channel 7:4 interrupt (C2IMR2[6:3])	0x0000 0068

Table 62. CPU2 vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
11	14	Settable	DMA2_CH[7:1] DMAMUX1_OVR	DMA2 channel 7:1 interrupt (C2IMR2[14:8]) DMAMUX1 overrun interrupt (C2IMR2[15])	0x0000 006C
12	15	Settable	ADC1 COMP	ADC1 global interrupt (C2IMR1[12]) COMP1 and COMP2 interrupt through EXTI[21:20] (C2IMR1[11])	0x0000 0070
13	16	Settable	LPTIM1	LP timer 1 global interrupt	0x0000 0074
14	17	Settable	LPTIM2	LP timer 2 global interrupt	0x0000 0078
15	18	Settable	TIM1_BRK, TIM1_UP, TIM1_TRG_COM, TIM1_CC	Timer 1 break, update, trigger and communication, capture compare interrupt	0x0000 007C
16	19	Settable	TIM2	Timer 2 global interrupt	0x0000 0080
17	20	Settable	TIM16	Timer 16 global interrupt	0x0000 0084
18	21	Settable	TIM17	Timer 17 global interrupt	0x0000 0088
19	22	Settable	IPCC_C2_RX_IT IPCC_C2_TX_IT HSEM	IPCC CPU2 RX occupied interrupt IPCC CPU2 TX free interrupt Semaphore interrupt 1 o CPU2	0x0000 008C
20	23	Settable	PKA True RNG AES1	Private key accelerator interrupt (C2IMR1[8]) True random number generator interrupt (C2IMR1[9]) AES1 global interrupt (C2IMR1[10])	0x0000 0090
21	24	Settable	AES2	AES2 global interrupt	0x0000 0094
22	25	Settable	LCD 802_IT1	LCD global interrupt (C2IMR2[22]) 802.15.4 interrupt 1	0x0000 0098
23	26	Settable	I2C1_EV I2C1_ER	I2C1 event interrupt I2C1 error interrupt	0x0000 009C
24	27	Settable	I2C3_EV I2C3_ERV	I2C3 event interrupt I2C3 error interrupt	0x0000 00A0
25	28	Settable	SPI1	SPI1 global interrupt	0x0000 00A4
26	29	Settable	SPI2	SPI2 global interrupt	0x0000 00A8
27	30	Settable	USART1	USART1 global interrupt	0x0000 00AC
28	31	Settable	LPUART1	LPUART global interrupt	0x0000 00B0
29	32	Settable	SAI1	SAI1 A and B global interrupt	0x0000 00B4
30	33	Settable	BLE_BLUE_IT BLE_RFC_IT BLE_RFFMS_IT BLE_HOST_WKUP	BLE blue controller interrupt BLE radio control interrupt BLE radio states interrupt BLE host wakeup interrupt	0x0000 00B8
31	34	Settable	802_IT2 802_HOST_WKUP	802.15.4 interrupt 2 802.15.4 host wakeup interrupt	0x0000 00BC

## 13.5 Interrupt list

The device wakeup sources are listed in [Table 63](#). Depending on its origin, the wakeup is handled according to different types, see [Section 14.5: EXTI functional behavior](#) for more information.

Some wakeup sources are able to generate an event to the CPUs. see [Event](#) column.

The wakeup source capability to wakeup CPU1 and or CPU2 is listed in [Wakeup](#) column.

For CPUs interrupt handling see [Section 13: Nested vectored interrupt controller \(NVIC\)](#).

For Wakeup handling see [Section 14: Extended interrupt and event controller \(EXTI\)](#).

**Table 63. Wakeup interrupt table**

EXTI no	Acronym	Description	EXTI type	Event	Wakeup
0	EXTI[0]	EXTI line 0 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
1	EXTI[1]	EXTI line 1 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
2	EXTI[2]	EXTI line 2 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
3	EXTI[3]	EXTI line 3 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
4	EXTI[4]	EXTI line 4 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
5	EXTI[5]	EXTI line 5 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
6	EXTI[6]	EXTI line 6 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
7	EXTI[7]	EXTI line 7 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
8	EXTI[8]	EXTI line 8 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
9	EXTI[8]	EXTI line 9 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
10	EXTI[10]	EXTI line 10 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
11	EXTI[11]	EXTI line 11 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
12	EXTI[12]	EXTI line 12 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
13	EXTI[13]	EXTI line 13 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
14	EXTI[14]	EXTI line 14 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
15	EXTI[15]	EXTI line 15 from SYSCFG	Configurable A	Yes	CPU1 and CPU2
16	PVD	PVD line	Configurable A	No	CPU1 and CPU2
17	RTC_ALARM	RTC Alarms (A and B) interrupt	Configurable A	Yes	CPU1 and CPU2
18	TAMP, RTC_STAMP, LSE_CSS	RTC Tamper interrupt RTC TimeStamp interrupt RCC LSECSS interrupt	Configurable A	Yes	CPU1 and CPU2
19	RTC_WKUP	RTC wakeup interrupt	Configurable A	Yes	CPU1 and CPU2
20	COMP1	COMP1 line	Configurable A	Yes	CPU1 and CPU2
21	COMP2	COMP2 line	Configurable A	Yes	CPU1 and CPU2
22	I2C1 wakeup	I2C1 wakeup	Direct B	No	CPU1 and CPU2
23	I2C3 wakeup	I2C3 wakeup	Direct B	No	CPU1 and CPU2

Table 63. Wakeup interrupt table (continued)

EXTI no.	Acronym	Description	EXTI type	Event	Wakeup
24	USART1	USART1 wakeup	Direct B	No	CPU1 and CPU2
25	LPUART1	LPUART1 wakeup	Direct B	No	CPU1 and CPU2
26	Reserved	-	-	-	-
27	Reserved	-	-	-	-
28	USB wakeup	USB wakeup	Direct B	No	CPU1 and CPU2
29	LPTIM1 wakeup	LP timer 1 wakeup	Direct B	No	CPU1 and CPU2
30	LPTIM2 wakeup	LP timer 2 wakeup	Direct B	No	CPU1 and CPU2
31	PVM1	PVM1 line	Configurable A	No	CPU1 and CPU2
32	Reserved	-	-	-	-
33	PVM3	PVM3 line	Configurable A	No	CPU1 and CPU2
34	Reserved	-	-	-	-
35	Reserved	-	-	-	-
36	IPCC CPU1 interrupts	IPCC CPU1 RX occupied and TX free interrupts	Direct C	No	CPU1 <sup>(1)</sup>
37	IPCC CPU2 interrupts	IPCC CPU2 RX occupied and TX free interrupts	Direct C	No	CPU2 <sup>(2)</sup>
38	HSEM interrupt 0	Semaphore interrupt 0 for CPU1	Direct C	No	CPU1 <sup>(1)</sup>
39	HSEM interrupt 1	Semaphore interrupt 1 for CPU2	Direct C	No	CPU2 <sup>(2)</sup>
40	C2SEV	CPU2 SEV line	Configurable A	Yes	CPU1 <sup>(3)</sup>
41	C1SEV	CPU1 SEV line	Configurable A	Yes	CPU2 <sup>(4)</sup>
42	Flash interrupt	Flash ECC and global interrupts	Direct C	No	CPU1 and CPU2
43	LCD wakeup	LCD wakeup	Direct B	No	CPU1 and CPU2
44	HSE CSS interrupt	RCC HSE CSS interrupt	Direct C	No	CPU1 and CPU2
45	BLE interrupts	BLE, RADIO, & RF_FSM interrupts	Direct B	No	CPU2 <sup>(2)</sup>
46	802 interrupts	802.15.4 Int0 and Int1 interrupts	Direct B	No	CPU2 <sup>(2)</sup>
47	Reserved	-	-	-	-
48	CDBGPOWERUPREQ	Debug power up request wakeup	Direct	No	CPU1 and CPU2

1. For correct operation the EXTI direct event C2IMRx.IMn bit must be set to 0 before CPU1 uses this direct event.
2. For correct operation the EXTI direct event C1IMRx.IMn bit must be set to 0 before CPU2 uses this direct event.
3. For correct operation the EXTI configurable event both C2IMRx.IMn and C2EMRx.EMn bits must be set to 0 before CPU1 uses this configurable event.
4. For correct operation the EXTI configurable event both C1IMRx.IMn and C1EMRx.EMn bits must be set to 0 before CPU2 uses this configurable event.

**14****Extended interrupt and event controller (EXTI)**

The Extended interrupt and event controller (EXTI) manages the individual CPU and system wakeup through configurable and direct event inputs. It provides wakeup requests to the power control, and generates an interrupt request to the CPUs interrupt controller and events to the CPUs event input. For each CPU an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wakeup requests allow the system to be woken up from STOP modes, and the CPU to be woken up from the CSTOP and CSTANDBY modes.

The interrupt request and event request generation can also be used in RUN modes.

**14.1****EXTI main features**

The EXTI main features are the following:

- 49 input events supported
- Two CPUs supported
- All event inputs allow to wake up the system
- Events which do not have an associated wakeup flag in the peripheral, have a flag in the EXTI and generate an interrupt to the CPU from the EXTI
- Some events can be used to generate a CPU wakeup event

The asynchronous event inputs are classified in two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse)
  - Configurable events have the following features:
    - Selectable active trigger edge
    - Interrupt pending status register bit.
    - Individual interrupt and event generation mask, used for conditioning the CPU wakeup, interrupt and event generation
    - SW trigger possibility
- Direct events (interrupt and wakeup sources from peripherals having an associated flag which requiring to be cleared in the peripheral)
  - Direct events have the following features:
    - Fixed rising edge active trigger
    - No interrupt pending status register bit in the EXTI (the interrupt pending status flag is provided by the peripheral generating the event)
    - Individual interrupt and event generation mask, used for conditioning the CPU wakeup and event generation
    - No SW trigger possibility

**14.2****EXTI implementation****Table 64. EXTI implementation**

Feature	STM32WB55xx	STM32WB35xx
Input events	0 to 48	0 to 42, 44 to 48

## 14.3 EXTI block diagram

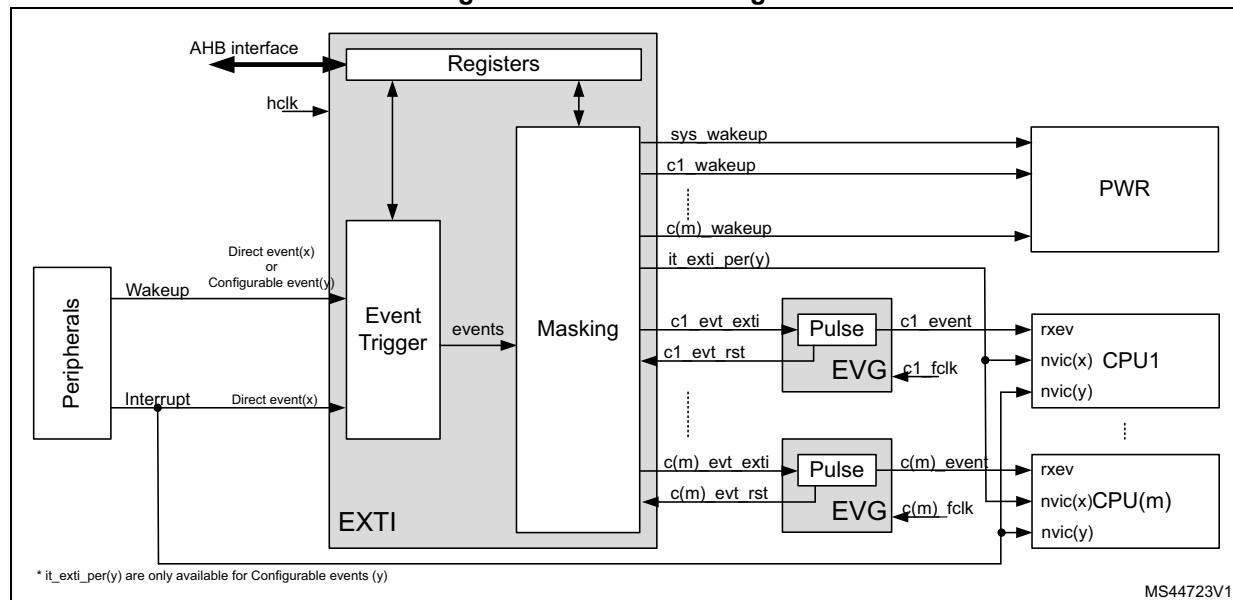
The EXTI consists of a register block accessed via an AHB interface, the event input Trigger block, and the masking block as shown in [Figure 30](#).

The register block contains all the EXTI registers.

The event input trigger block provides event input edge trigger logic.

The masking block provides the event input distribution to the different wakeup, interrupt and event outputs, and the masking of these.

**Figure 30. EXTI block diagram**



**Table 65. EXTI pin overview**

Pin name	I/O	Description
AHB interface	I/O	EXTI register bus interface. When one event is configured to allow security, the AHB interface supports secure accesses.
hclk	I	AHB bus clock and EXTI system clock.
Configurable event(y)	I	Asynchronous wakeup events from peripherals that do not have an associated interrupt and flag in the peripheral.
Direct event(x)	I	Synchronous and asynchronous wakeup events from peripherals having an associated interrupt and flag in the peripheral.
<code>it_exti_per(y)</code>	O	Interrupts to the CPU1 to CPU(m) associated with Configurable event (y).
<code>c(m)_evt_exti</code>	O	High level sensitive event output for CPU(m) synchronous to hclk. (m= 1 to 2)
<code>c(m)_evt_rst</code>	I	Asynchronous reset input to clear <code>c(m)_evt_exti</code> . (m= 1 to 2)
<code>sys_wakeup</code>	O	Asynchronous system wakeup request to PWR for <code>ck_sys</code> and <code>hclk</code> .
<code>c(m)_wakeup</code>	O	Wakeup request to PWR for CPU(m), synchronous to <code>hclk</code> . (m= 1 to 2)

**Table 66. EVG pin overview**

Pin name	I/O	Description
c_fclk	I	CPU free running clock.
c_evt_in	I	High level sensitive Events input from EXTI, asynchronous to CPU clock.
c_event	O	Event pulse, synchronous to CPU clock.
c_evt_rst	O	Event reset signal, synchronous to CPU clock.

### 14.3.1 EXTI connections between peripherals and CPU

The peripherals able to generate wakeup or interrupt events when the system is in STOP mode or a CPU is in CSTOP mode are connected to the EXTI.

- Peripheral wakeup signals that generate a pulse or which do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events the EXTI provides a status pending bit that has to be cleared. It is the EXTI interrupt associated with the status bit that will interrupt the CPU.
- Peripheral interrupt and wakeup signals that have a status bit in the peripheral that has to be cleared in the peripheral, are connected to an EXTI direct event input. There is no status pending bit within the EXTI. The interrupt or wakeup is cleared by the CPU in the peripheral. It is the peripheral interrupt that will interrupt the CPU directly.

The EXTI configurable event interrupts are connected to the respective interrupt controller of each CPU(m).

The dedicated EXTI/EVG CPU(m) event is connected to the respective CPU(m) rxev input.

The EXTI CPU(m) wakeup signals are connected to the PWR block, and are used to wake up the system and CPU(m) sub-system bus clocks.

## 14.4 EXTI functional description

Depending on the EXTI event input type and wakeup target(s), different logic implementations are used. The applicable features are controlled from register bits:

- Active trigger edge enable, by rising edge selection  
*EXTI rising trigger selection register (EXTI\_RTSR1),*  
*EXTI rising trigger selection register (EXTI\_RTSR2),*  
and falling edge selection  
*EXTI falling trigger selection register (EXTI\_FTSR1),*  
*EXTI falling trigger selection register (EXTI\_FTSR2).*
- Software trigger, by  
*EXTI software interrupt event register (EXTI\_SWIER1),*  
*EXTI software interrupt event register (EXTI\_SWIER2).*
- Interrupt pending flag, by  
*EXTI pending register (EXTI\_PR1),*  
*EXTI pending register (EXTI\_PR2).*
- CPU wakeup and interrupt enable, by  
*EXTI CPU wakeup with interrupt mask register (EXTI\_IMR1),*  
*EXTI CPU2 wakeup with interrupt mask register (EXTI\_C2IMR1),*  
*EXTI CPU wakeup with interrupt mask register (EXTI\_IMR2),*  
*EXTI CPU2 wakeup with interrupt mask register (EXTI\_C2IMR2).*
- CPU wakeup and event enable, by  
*EXTI CPU wakeup with event mask register (EXTI\_EMR1),*  
*EXTI CPU2 wakeup with event mask register (EXTI\_C2EMR1)*  
*EXTI CPU wakeup with event mask register (EXTI\_EMR2),*  
*EXTI CPU2 wakeup with event mask register (EXTI\_C2EMR2).*

**Table 67. EXTI event input configurations and register control**

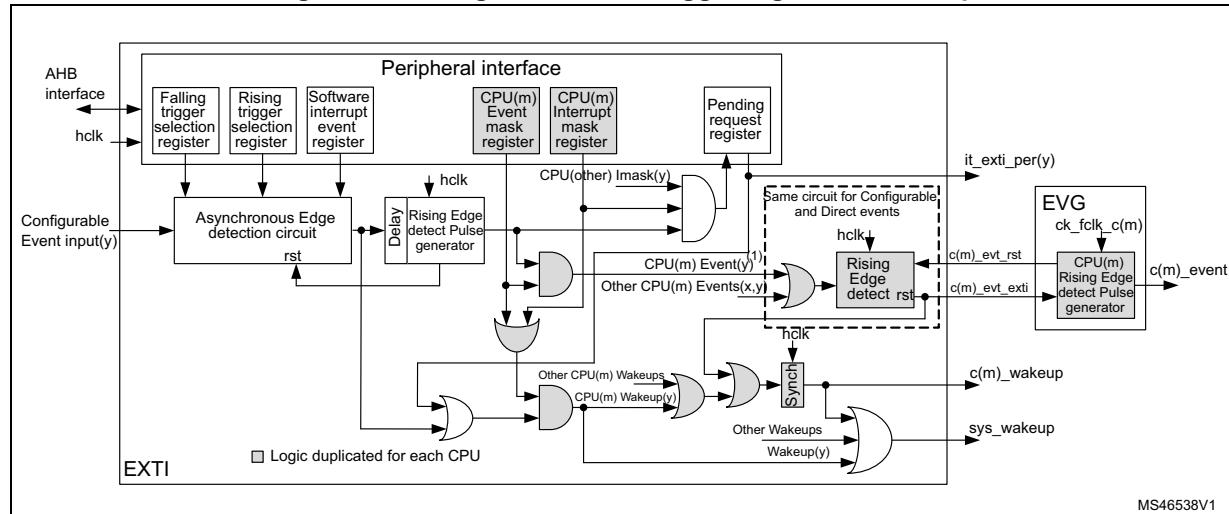
Event input type	Logic implementation	EXTI_RTSR	EXTI_FTSR	EXTI_SWIER	EXTI_PR	EXTI_CmIMR	EXTI_CmEMR <sup>(1)</sup>
Configurable	Configurable event input wakeup logic	x	x	x	x	x	x
Direct	Direct event input wakeup logic	-	-	-	-	x	x

1. Only for input events with configuration “rxev generation” enabled.

### 14.4.1 EXTI configurable event input wakeup

*Figure 31* is a detailed representation of the logic associated with configurable event inputs which will wake up the CPU(m) sub-system bus clocks and generated an EXTI pending flag and interrupt to the CPU(m) and or a CPU(m) wakeup event.

Figure 31. Configurable event trigger logic CPU wakeup



1. Only for the input events that support CPU rxev generation  $c(n)_event$ .

The software interrupt event register allows to trigger configurable events by software, writing the corresponding register bit, irrespective of the edge selection setting.

The rising edge and falling edge selection registers allow to enable and select the configurable event active trigger edge or both edges.

Each CPU has its dedicated interrupt mask register and a dedicated event mask registers. The enabled event allows to generate an event on the CPU. All events for a CPU are OR-ed together into a single CPU event signal. The event pending register (EXTI\_PR) is not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers, shared by the CPUs. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to all CPUs. The configurable event interrupts need to be acknowledged by software in the EXTI\_PR register.

When a CPU(m) interrupt or CPU(m) event is enabled the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

**Note:** *A detected configurable event interrupt pending request, may be cleared by any CPU. The system is unable to enter into Low-power modes as long as an interrupt pending request is active.*

#### 14.4.2 EXTI direct event input wakeup

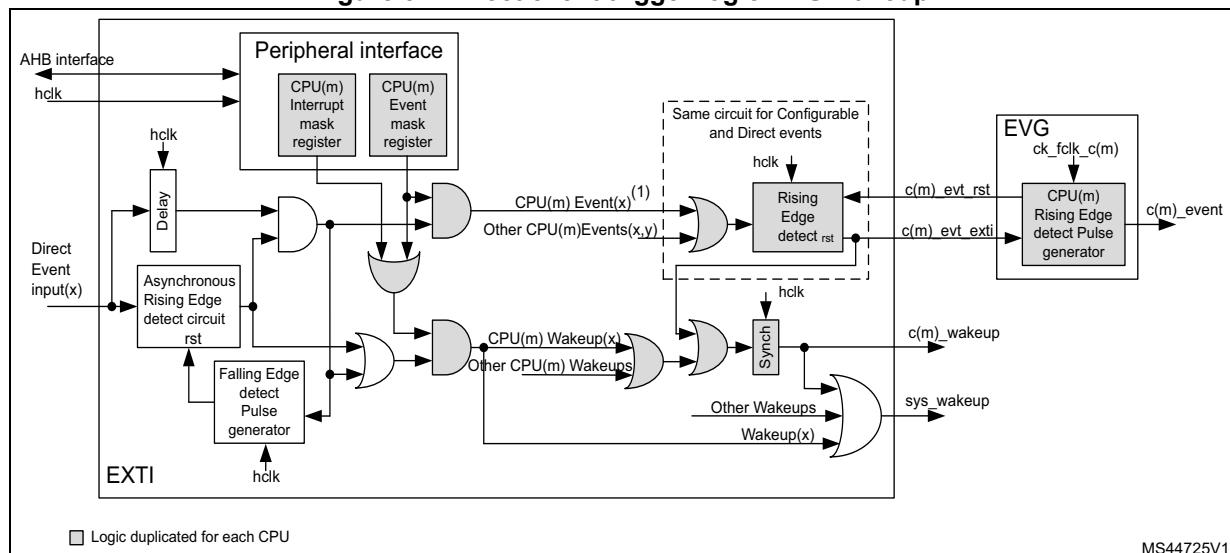
*Figure 32* is a detailed representation of the logic associated with direct event inputs waking up the system.

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU sub-system clocks and may generate a CPU wakeup event. The peripheral synchronous interrupt, associated with the direct wakeup event wakes up the CPU.

The EXTI direct event is able to generate a CPU event. This CPU event wakes up the CPU. The CPU event may occur before the associated Peripheral interrupt flag is set.

**Note:** *The direct events are cleared in the peripheral generating the event. When a direct event input enabled by CPU(m) is cleared by another CPU before the CPU(m) clock is running, the CPU(m) no longer receives a CPU(m) interrupt nor a CPU(m) event and does not wake up. However the system stays in RUN mode, generating the CPU(m) clock. For this reason CPU(m) direct events must NOT be cleared by the other CPU.*

**Figure 32. Direct event trigger logic CPU wakeup**



1. Only for the input events that support CPU rxev generation  $c(n)_event$ .

#### 14.5 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the wakeup event. The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the generation of a CPU(m) wakeup is conditioned by the CPU(m) interrupt mask and CPU(m) event mask.

Table 68. Masking functionality

CPU interrupt enable EXTI_CmIMR.IMn	CPU event enable EXTI_CmEMR.EMn	Configurable event inputs EXTI_PR.PIFn	exti(n) interrupt <sup>(1)</sup>	CPU(m) event	CPU(m) wakeup
0 for all CPUs	0	No	Masked	Masked	Masked
	1		Masked	Yes	Yes
1 for any CPU	0	Status latched	Yes	Masked	Yes <sup>(2)</sup>
	1		Yes	Yes	Yes

1. The single exti(n) interrupt goes to all CPUs. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in the CPU(m) interrupt controller.

2. Only if CPU(m) interrupt is enabled in EXTI\_CmIMR.IMn.

For configurable event inputs, when the enabled edge(s) occur on the event input, an event request is generated. When the associated CPU(m) interrupt is unmasked the corresponding pending bit EXTI\_PR.PIFn is set and the CPU(m) sub-system is woken up and CPU interrupt signal is activated. The EXTI\_PR.PIFn pending bit must be cleared by software writing it to '1'. This clears the CPU interrupt.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit in the EXTI. When the associated CPU(m) interrupt is unmasked the corresponding CPU sub-system is woken up. The CPU is woken up (interrupted) by the peripheral synchronous interrupt.

The CPU(m) event must be unmasked to generate an event. When the enabled edge(s) occur on the event input a CPU(m) event pulse is generated. There is no event pending bit.

For the configurable event inputs an event request can be generated by software when writing a '1' in the software interrupt/event register EXTI\_SWIER, generating a rising edge on the event. The edge event pending bit is set in EXTI\_PR, irrespective of the setting in EXTI\_RTSR.

## 14.6 EXTI registers

The EXTI register map is divided as detailed in [Table 69](#).

**Table 69. EXTI register map sections**

Address	Description
0x000 - 0x01C	General configurable event [31:0] configuration
0x020 - 0x03C	General configurable event [63:32] configuration
0x040 - 0x05C	General configurable event [95:64] configuration
0x080 - 0x0BC	CPU1 input event configuration
0x0C0 - 0x0FC	CPU2 input event configuration

All the registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

### 14.6.1 EXTI rising trigger selection register (EXTI\_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RT31	Res.	RT21	RT20	RT19	RT18	RT17	RT16								
rw										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw															

Bit 31 **RT31**: Rising trigger event configuration bit of configurable Event input 31<sup>(1)</sup>.

- 0: Rising trigger disabled (for event and Interrupt) for input line
- 1: Rising trigger enabled (for event and Interrupt) for input line

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:0 **RT[21:0]**: Rising trigger event configuration bit of configurable event input x (x = 21 to 0)<sup>(1)</sup>.

- 0: Rising trigger disabled (for event and Interrupt) for input line
- 1: Rising trigger enabled (for event and Interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

### 14.6.2 EXTI falling trigger selection register (EXTI\_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FT31	Res.	FT21	FT20	FT19	FT18	FT17	FT16								
rw										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw															

Bit 31 **FT31**: Falling trigger event configuration bit of configurable event input 31<sup>(1)</sup>.

- 0: Falling trigger disabled (for event and Interrupt) for input line
- 1: Falling trigger enabled (for event and Interrupt) for input line.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:0 **FT[21:0]**: Falling trigger event configuration bit of configurable event input x (x = 21 to 0)<sup>(1)</sup>.

- 0: Falling trigger disabled (for event and Interrupt) for input line
- 1: Falling trigger enabled (for event and Interrupt) for input line.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

### 14.6.3 EXTI software interrupt event register (EXTI\_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWI31	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI21	SWI20	SWI19	SWI18	SWI17	SWI16
rw										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SWI31**: Software interrupt on event 31

A software interrupt is generated independent from the setting in EXTI\_RTSR and EXTI\_FTSR. Will always return 0 when read.

- 0: Writing 0 has no effect.
- 1: Writing a 1 to this bit will trigger a rising edge event on event 31.

This bit is auto cleared by HW.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:0 **SWI[21:0]**: Software interrupt on event x (x = 21 to 0)

A software interrupt is generated independent from the setting in EXTI\_RTSR and EXTI\_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event x.

This bit is auto cleared by HW.

#### 14.6.4 EXTI pending register (EXTI\_PR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIF31	Res.	PIF21	PIF20	PIF19	PIF18	PIF17	PIF16								
rc_w1										rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIF15	PIF14	PIF13	PIF12	PIF11	PIF10	PIF9	PIF8	PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
rc_w1															

Bit 31 **PIF31**: configurable event inputs 31 Pending bit.

0: No trigger request occurred

1: Selected trigger request occurred

This bit is set when the selected edge event or an EXTI\_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:0 **PIF[21:0]**: configurable event inputs x (x = 21 to 0) Pending bit.

0: No trigger request occurred

1: Selected trigger request occurred

This bit is set when the selected edge event or an EXTI\_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

#### 14.6.5 EXTI rising trigger selection register (EXTI\_RTSR2)

Address offset: 0x020

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RT41	RT40	Res.	Res.	Res.	Res.	Res.	Res.	RT33	Res.
						rw	rw								rw

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

Bit 9 **RT41**: Rising trigger event configuration bit of configurable event input 41<sup>(1)</sup>

- 0: Rising trigger disabled (for event and interrupt) for input line
- 1: Rising trigger enabled (for event and interrupt) for input line

Bit 8 **RT40**: Rising trigger event configuration bit of configurable event input 40<sup>(1)</sup>

- 0: Rising trigger disabled (for event and interrupt) for input line
- 1: Rising trigger enabled (for event and interrupt) for input line

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **RT33**: Rising trigger event configuration bit of configurable event input 33<sup>(1)</sup>

- 0: Rising trigger disabled (for event and interrupt) for input line
- 1: Rising trigger enabled (for event and interrupt) for input line

Bit 0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable Event input. In this case, both edges generate a trigger.

#### 14.6.6 EXTI falling trigger selection register (EXTI\_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FT41	FT40	Res.	Res.	Res.	Res.	Res.	Res.	FT33	Res.
						rw	rw								rw

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

Bit 9 **FT41**: Falling trigger event configuration bit of configurable event input 41<sup>(1)</sup>

- 0: Falling trigger disabled (for event and interrupt) for input line
- 1: Falling trigger enabled (for event and interrupt) for input line

Bit 8 **FT40**: Falling trigger event configuration bit of configurable event input 40<sup>(1)</sup>

- 0: Falling trigger disabled (for event and interrupt) for input line
- 1: Falling trigger enabled (for event and interrupt) for input line

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **FT33**: Falling trigger event configuration bit of configurable event input 33<sup>(1)</sup>

- 0: Falling trigger disabled (for event and interrupt) for input line
- 1: Falling trigger enabled (for event and interrupt) for input line

Bit 0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

### 14.6.7 EXTI software interrupt event register (EXTI\_SWIER2)

Address offset: 0x028

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SWI41	SWI40	Res.	Res.	Res.	Res.	Res.	Res.	SWI33	Res.
						rw	rw								rw

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

Bit 9 **SWI41**: Software interrupt on event 41

A software interrupt is generated independent from the setting in EXTI\_RTSR and EXTI\_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event 41.

This bit is auto cleared by HW.

Bit 8 **SWI40**: Software interrupt on event 40

A software interrupt is generated independent from the setting in EXTI\_RTSR and EXTI\_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event 40.

This bit is auto cleared by HW.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **SWI33**: Software interrupt on event 33

A software interrupt is generated independent from the setting in EXTI\_RTSR and EXTI\_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event 33.

This bit is auto cleared by HW.

Bit 0 Reserved, must be kept at reset value.

### 14.6.8 EXTI pending register (EXTI\_PR2)

Address offset: 0x02C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	PIF41	PIF40	Res.	Res.	Res.	Res.	Res.	Res.	PIF33	Res.
						rc_w1	rc_w1								rc_w1

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

Bit 9 **PIF41**: configurable event inputs 41 pending bit

0: No trigger request occurred

1: selected trigger request occurred

This bit is set when the selected edge event or an EXTI\_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 8 **PIF40**: configurable event inputs 40 pending bit

0: No trigger request occurred

1: selected trigger request occurred

This bit is set when the selected edge event or an EXTI\_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **PIF33**: configurable event inputs 33 pending bit

0: No trigger request occurred

1: selected trigger request occurred

This bit is set when the selected edge event or an EXTI\_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 0 Reserved, must be kept at reset value.

### 14.6.9 EXTI CPU wakeup with interrupt mask register (EXTI\_IMR1)

Address offset: 0x080

Reset value: 0x7FC0 0000

Contains register bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw															

Bits 31:0 **IM[31:0]**: CPU wakeup with interrupt mask on event input x (x = 31:0)<sup>(1)(2)</sup>.

0: Wakeup with interrupt request from Line x is masked

1: Wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to '0' in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to '1' in order to enable the interrupt by default.

#### 14.6.10 EXTI CPU2 wakeup with interrupt mask register (EXTI\_C2IMR1)

Address offset: 0x0C0

Reset value: 0x7FC0 0000

Contains register bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw															

Bits 31:0 **IM[31 :0]**: CPU2 wakeup with interrupt mask on event input x (x = 31 to 0)<sup>(1)(2)</sup>.

0: Wakeup with interrupt request from Line x is masked

1: Wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to '0' in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to '1' in order to enable the interrupt by default.

#### 14.6.11 EXTI CPU wakeup with event mask register (EXTI\_EMR1)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	EM21	EM20	EM19	EM18	EM17	Res.									
										rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw															

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:17 **EM[21:17]**: CPU wakeup with event generation mask on event input x (x = 21 to 17)

0: Wakeup with event generation from Line x is masked

1: Wakeup with event generation from Line x is unmasked

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **EM[15:0]**: CPU wakeup with event generation mask on event input x (x = 15 to 0)

0: Wakeup with event generation from Line x is masked

1: Wakeup with event generation from Line x is unmasked

### 14.6.12 EXTI CPU2 wakeup with event mask register (EXTI\_C2EMR1)

Address offset: 0x0C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	EM21	EM20	EM19	EM18	EM17	Res.									
										rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw															

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:17 **EM[21:17]**: CPU2 wakeup with event generation mask on event input x (x = 21 to 17)

0: Wakeup with event generation from Line x is masked

1: Wakeup with event generation from Line x is unmasked

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **EM[15:0]**: CPU2 wakeup with event generation mask on event input x (x = 15 to 0)

0: Wakeup with event generation from Line x is masked

1: Wakeup with event generation from Line x is unmasked

### 14.6.13 EXTI CPU wakeup with interrupt mask register (EXTI\_IMR2)

Address offset: 0x090

Reset value: 0x0001 FCFD

Contains register bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	IM48														
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM47	IM46	IM45	IM44	IM43	IM42	IM41	IM40	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
rw															

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

Bits 16:0 **IM[48:32]**: CPU wakeup with interrupt mask on event input x (x = 48 to 32)<sup>(1)(2)</sup>

0: Wakeup with interrupt request from Line x is masked

1: Wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to '0' in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to '1' in order to enable the interrupt by default.

#### 14.6.14 EXTI CPU2 wakeup with interrupt mask register (EXTI\_C2IMR2)

Address offset: 0x0D0

Reset value: 0x0001 FCFD

Contains register bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	IM48														
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM47	IM46	IM45	IM44	IM43	IM42	IM41	IM40	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
rw															

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

Bits 16:0 **IM[48:32]**: CPU wakeup with interrupt mask on event input x (x = 48 to 32)<sup>(1)(2)</sup>

- 0: Wakeup with interrupt request from Line x is masked
- 1: Wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to '0' in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to '1' in order to enable the interrupt by default.

#### 14.6.15 EXTI CPU wakeup with event mask register (EXTI\_EMR2)

Address offset: 0x0D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EM41	EM40	Res.							
						rw	rw								

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

Bit 9 **EM41**: CPU wakeup with event generation mask on event input 41

- 0: Wakeup with event generation from Line 41 is masked
- 1: Wakeup with event generation from Line 41 is unmasked

Bit 8 **EM40**: CPU wakeup with event generation mask on event input 40.

- 0: Wakeup with event generation from Line 40 is masked
- 1: Wakeup with event generation from Line 40 is unmasked

Bits 7:0 Reserved, must be kept at reset value.

### 14.6.16 EXTI CPU2 wakeup with event mask register (EXTI\_C2EMR2)

Address offset: 0x0D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EM41	EM40	Res.							
						rw	rw								

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

Bit 9 **EM41**: CPU2 wakeup with event generation mask on event input 41

- 0: Wakeup with event generation from Line 41 is masked
- 1: Wakeup with event generation from Line 41 is unmasked

Bit 8 **EM40**: CPU2 wakeup with event generation mask on event input 40.

- 0: Wakeup with event generation from Line 40 is masked
- 1: Wakeup with event generation from Line 40 is unmasked

Bits 7:0 Reserved, must be kept at reset value.

## 14.6.17 EXTI register map

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

**Table 70. EXTI register map and reset values**

Offset	Register	RT[31]	31
0x000	<b>EXTI_RTSR1</b>	RT[31]	31
	Reset value	0	0
0x004	<b>EXTI_FTSR1</b>	FT[31]	30
	Reset value	0	0
0x008	<b>EXTI_SWIER1</b>	SWI[31]	29
	Reset value	0	0
0x00C	<b>EXTI_PR1</b>	PIF[31]	28
	Reset value	0	0
0x010-0x01C	Reserved	Res.	27
0x020	<b>EXTI_RTSR2</b>	Res.	26
	Reset value	0	0
0x024	<b>EXTI_FTSR2</b>	Res.	25
	Reset value	0	0
0x028	<b>EXTI_SWIER2</b>	Res.	24
	Reset value	0	0
0x02C	<b>EXTI_PR2</b>	Res.	23
	Reset value	0	0
0x030-0x07C	Reserved	Res.	22
0x080	<b>EXTI_IMR1</b>	IM[31:0]	21
	Reset value	0	0
0x084	<b>EXTI_EMR1</b>	EM[21:17]	20
	Reset value	0	0
0x088-0x08C	Reserved	Res.	19
0x090	<b>EXTI_IMR2</b>	IM[48:32]	18
	Reset value	0	0
0x094	<b>EXTI_EMR2</b>	EM[41:40]	17
	Reset value	0	0
0x098-0x0BC	Reserved	Res.	16
0x0C0	<b>EXTI_IMR3</b>	IM[33:0]	15
	Reset value	0	0
0x0C4	<b>EXTI_EMR3</b>	EM[41:40]	14
	Reset value	0	0
0x0C8	<b>EXTI_IMR4</b>	IM[33:0]	13
	Reset value	0	0
0x0CC	<b>EXTI_EMR4</b>	EM[41:40]	12
	Reset value	0	0
0x0D0	<b>EXTI_IMR5</b>	IM[33:0]	11
	Reset value	0	0
0x0D4	<b>EXTI_EMR5</b>	EM[41:40]	10
	Reset value	0	0
0x0D8	<b>EXTI_IMR6</b>	IM[33:0]	9
	Reset value	0	0
0x0DC	<b>EXTI_EMR6</b>	EM[41:40]	8
	Reset value	0	0
0x0E0	<b>EXTI_IMR7</b>	IM[33:0]	7
	Reset value	0	0
0x0E4	<b>EXTI_EMR7</b>	EM[41:40]	6
	Reset value	0	0
0x0E8	<b>EXTI_IMR8</b>	IM[33:0]	5
	Reset value	0	0
0x0F0	<b>EXTI_EMR8</b>	EM[41:40]	4
	Reset value	0	0
0x0F4	<b>EXTI_IMR9</b>	IM[33:0]	3
	Reset value	0	0
0x0F8	<b>EXTI_EMR9</b>	EM[41:40]	2
	Reset value	0	0
0x0F0	<b>EXTI_IMR10</b>	IM[33:0]	1
	Reset value	0	0

Table 70. EXTI 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
0x0C0	EXTI_C2IMR1																																
	Reset value	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C4	EXTI_C2EMR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	Reset value																																
0x0C8-0x0CC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																																
0x0D0	EXTI_C2IMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
	Reset value																																
0x0D4	EXTI_C2EMR2	Res.	EM [41:40]	0	0	Res.																											
	Reset value																																

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 15 Quad-SPI interface (QUADSPI)

### 15.1 Introduction

The QUADSPI is a specialized communication interface targeting single, dual- or quad-SPI flash memories. It can operate in any of the three following modes:

- indirect mode: all the operations are performed using the QUADSPI registers.
- automatic status-polling mode: the external flash memory status register is periodically read and an interrupt can be generated in case of flag setting.
- memory-mapped mode: the external flash memory is mapped to the device address space and is seen by the system as if it was an internal memory.

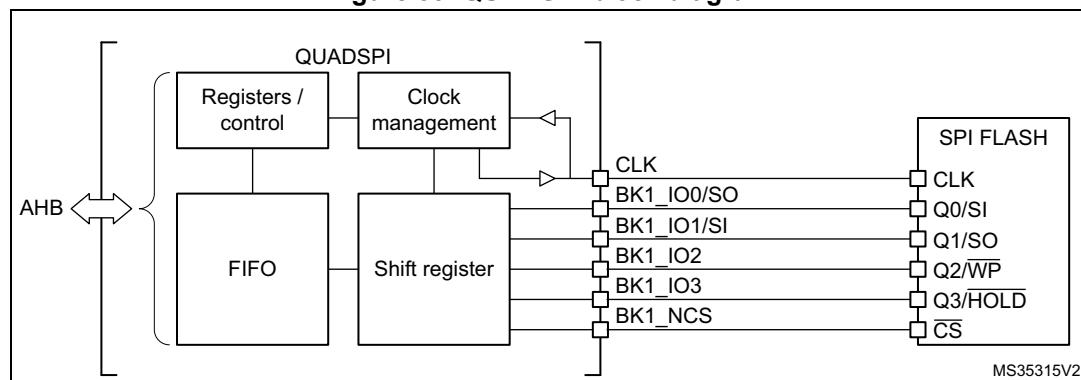
### 15.2 QUADSPI main features

- Three functional modes: indirect, automatic status-polling, and memory-mapped
- SDR and DDR support
- Fully programmable opcode for both indirect and memory-mapped modes
- Fully programmable frame format for both indirect and memory-mapped modes
- Integrated FIFO for reception and transmission
- 8-, 16-, and 32-bit data accesses allowed
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error

### 15.3 QUADSPI functional description

#### 15.3.1 QUADSPI block diagram

Figure 33. QUADSPI block diagram



### 15.3.2 QUADSPI pins

The table below lists the QUADSPI pins for interfacing with a flash memory.

Table 71. QUADSPI pins

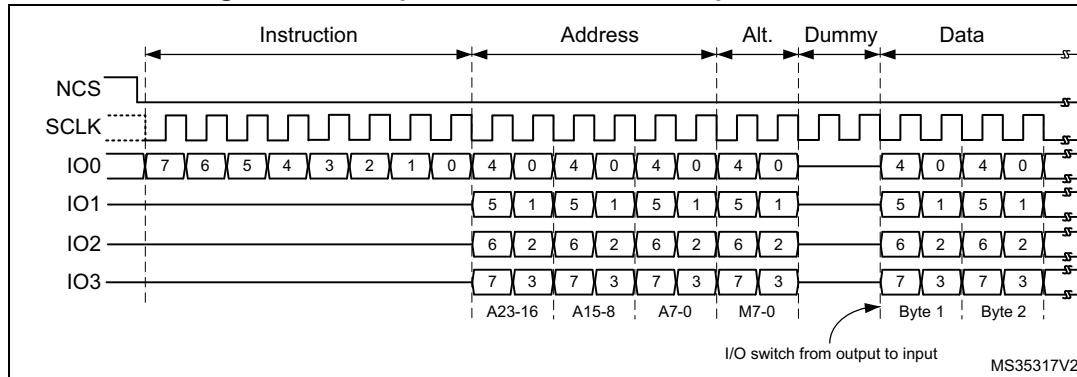
Signal name	Signal type	Description
CLK	Digital output	Clock to flash memory
BK1_IO0/SO	Digital input/output	Bidirectional I/O in dual/quad modes or serial output in single mode
BK1_IO1/SI	Digital input/output	Bidirectional I/O in dual/quad modes or serial input in single mode
BK1_IO2	Digital input/output	Bidirectional I/O in quad mode
BK1_IO3	Digital input/output	Bidirectional I/O in quad mode
BK1_NCS	Digital output	Chip select (active low)

### 15.3.3 QUADSPI command sequence

The QUADSPI communicates with the flash memory using commands. Each command can include five phases: instruction, address, alternate byte, dummy, data. Any of these phases can be configured to be skipped, but at least one of the instruction, address, alternate byte, or data phase must be present.

NCS falls before the start of each command and rises again after each command finishes.

Figure 34. Example of read command in quad-SPI mode



#### Instruction phase

During this phase, an 8-bit instruction, configured in INSTRUCTION field of QUADSPI\_CCR[7:0] register, is sent to the flash memory, specifying the type of operation to be performed.

Most flash memories can receive instructions only one bit at a time from the IO0/SO signal (single-SPI mode), the instruction phase can optionally send 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the IMODE[1:0] field of QUADSPI\_CCR[9:8] register.

When IMODE = 00, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

### Address phase

In the address phase, 1-4 bytes are sent to the flash memory to indicate the address of the operation. The number of address bytes to be sent is configured in the ADSIZE[1:0] field of QUADSPI\_CCR[13:12] register. In indirect and automatic status-polling modes, address bytes to be sent are specified in the ADDRESS[31:0] field of QUADSPI\_AR register, while in memory-mapped mode, the address is given directly via the AHB (from the Cortex or from a DMA).

The address phase can send 1 bit at a time (over SO in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ADMODE[1:0] field of QUADSPI\_CCR[11:10] register.

When ADMODE = 00, the address phase is skipped, and the command sequence proceeds directly to the next phase, if any.

### Alternate-byte phase

In the alternate-byte phase, 1-4 bytes are sent to the flash memory, generally to control the mode of operation. The number of alternate bytes to be sent is configured in the [1:0] field of QUADSPI\_CCR[17:16] register. The bytes to be sent are specified in the QUADSPI\_ABR register.

The alternate-bytes phase can send 1 bit at a time (over SO in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ABMODE[1:0] field of QUADSPI\_CCR[15:14] register.

When ABMODE = 00, the alternate-byte phase is skipped, and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-mode is used and only two cycles are used for the alternate bytes. In this case, the firmware can use quad-mode (ABMODE = 11) and send a byte with bits 7 and 3 of ALTERNATE set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low). In this case, the upper two bits of the nibble to be sent are placed in bits 4:3 of ALTERNATE, while the lower two bits are placed in bits 1 and 0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE must be set to 0x8A (1000\_1010).

### Dummy-cycle phase

In the dummy-cycle phase, 1-31 cycles are given without any data being sent or received, in order to give time to the flash memory to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified in the DCYC[4:0] field of QUADSPI\_CCR[22:18] register. In both SDR and DDR modes, the duration is specified as a number of full CLK cycles.

When DCYC is zero, the dummy-cycles phase is skipped, and the command sequence proceeds directly to the data phase, if present.

The operating mode of the dummy-cycles phase is determined by DMODE.

In order to assure enough “turn-around” time for changing data signals from output mode to input mode, there must be at least one dummy cycle when using dual or quad mode to receive data from the flash memory.

### Data phase

During the data phase, any number of bytes can be sent to, or received from the flash memory.

In indirect and automatic status-polling modes, the number of bytes to be sent/received is specified in the QUADSPI\_DLR register.

In indirect-write mode the data to be sent to the flash memory must be written to the QUADSPI\_DR register. In indirect-read mode the data received from the flash memory is obtained by reading the QUADSPI\_DR register.

In memory-mapped mode, the data which is read is sent back directly over the AHB to the Cortex or to a DMA.

The data phase can send/receive 1 bit at a time (over SO/SI in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ABMODE[1:0] field of QUADSPI\_CCR[15:14] register.

When DMODE = 00, the data phase is skipped, and the command sequence finishes immediately by raising NCS. This configuration must only be used in only indirect write mode.

## 15.3.4 QUADSPI signal interface protocol modes

### Single-SPI mode

This legacy SPI mode allows just one single bit to be sent/received serially. In this mode, data are sent to the flash memory over the SO signal (whose I/O shared with IO0). Data received from the flash memory arrive via SI (whose I/O shared with IO1).

The different phases can each be configured separately to use this mode by setting the IMODE/ADMODE/ABMODE/DMODE fields (in QUADSPI\_CCR) to 01.

In each phase which is configured in single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).

This is the case even for the dummy phase if DMODE = 01.

### Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields of QUADSPI\_CCR register to 10.

In each phase which is configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.

In the dummy phase when DMODE = 01, IO0/IO1 are always high-impedance.

### Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use quad-SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields of QUADSPI\_CCR register to 11.

In each phase which is configured in this mode, IO0/IO1/IO2/IO3 are all are at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE = 11, IO0/IO1/IO2/IO3 are all high-impedance.

IO2 and IO3 are used only in quad-SPI mode. If none of the phases are configured to use quad-SPI mode, then the pins corresponding to IO2 and IO3 can be used for other functions even while the QUADSPI is active.

### SDR mode

By default, the DDRM bit (QUADSPI\_CCR[31]) is 0 and the QUADSPI operates in single-data rate (SDR) mode.

In SDR mode, when the QUADSPI drives IO0/SO, IO1, IO2, IO3 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the QUADSPI assumes that flash memories also send the data using CLK falling edge. By default (when SSHIFT = 0), the signals are sampled using the following (rising) edge of CLK.

### DDR mode

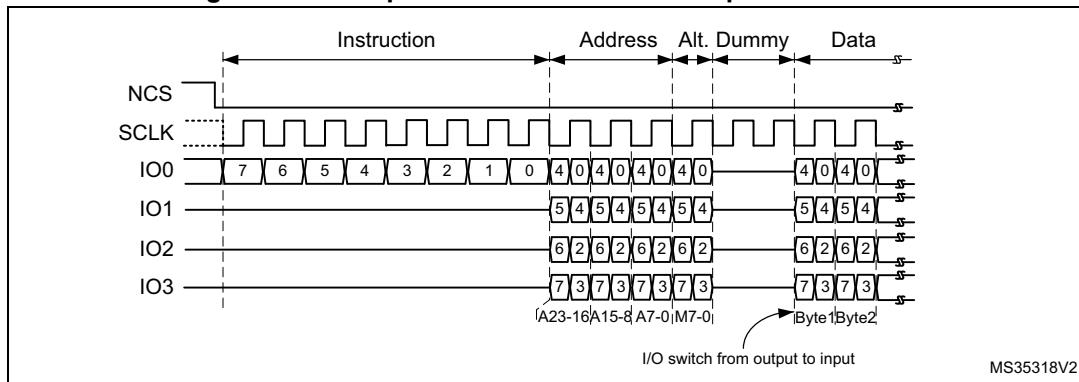
When the DDRM bit (QUADSPI\_CCR[31]) is set to 1, the QUADSPI operates in double-data rate (DDR) mode.

In DDR mode, when the QUADSPI is driving the IO0/SO, IO1, IO2, IO3 signals in the address/alternate-byte/data phases, a bit is sent on each of CLK falling and rising edges.

The instruction phase is not affected by DDRM. The instruction is always sent using CLK falling edge.

When receiving data in DDR mode, the QUADSPI assumes that flash memories also send the data using both CLK rising and falling edges. When DDRM = 1, the firmware must clear SSHIFT (bit 4 of QUADSPI\_CR). Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

Figure 35. Example of a DDR command in quad-SPI mode



### 15.3.5 QUADSPI indirect mode

When in indirect mode, commands are started by writing to QUADSPI registers, and data are transferred by writing or reading the data register, in the same way as for other communication peripherals.

When FMODE = 00 (QUADSPI\_CCR[27:26]), the QUADSPI is in indirect-write mode, where bytes are sent to the flash memory during the data phase. Data are provided by writing to the data register (QUADSPI\_DR).

When FMODE = 01, the QUADSPI is in indirect-read mode, where bytes are received from the flash memory during the data phase. Data are recovered by reading QUADSPI\_DR.

The number of bytes to be read/written is specified in the data length register (QUADSPI\_DLR). If QUADSPI\_DLR = 0xFFFF\_FFFF (all 1s), then the data length is considered undefined and the QUADSPI simply continues to transfer data until the end of flash memory (as defined by FSIZE) is reached. If no bytes are to be transferred, DMODE (QUADSPI\_CCR[25:24]) must be set to 00.

If QUADSPI\_DLR = 0xFFFF\_FFFF and FSIZE = 0x1F (max value indicating a 4-Gbyte flash memory), then in this special case, transfers continue indefinitely, stopping only after an abort request or after the QUADSPI is disabled. After the last memory address is read (at address 0xFFFF\_FFFF), reading continues with address = 0x0000\_0000.

When the programmed number of bytes to be transmitted or received is reached, TCF is set and an interrupt is generated if TCIE = 1. In the case of undefined number of data, the TCF is set when the limit of the external SPI memory is reached according to the flash memory size defined in QUADSPI\_CR.

### Triggering the start of a command

Essentially, a command starts as soon as firmware gives the last information that is necessary for this command. Depending on the QUADSPI configuration, there are three different ways to trigger the start of a command in indirect mode. The command starts immediately:

- after a write is performed to INSTRUCTION[7:0] (QUADSPI\_CCR), if no address is necessary (when ADMODE = 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
- after a write is performed to ADDRESS[31:0] (QUADSPI\_AR), if an address is necessary (when ADMODE ≠ 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
- after a write is performed to DATA[31:0] (QUADSPI\_DR), if data needs to be provided by the firmware (when FMODE = 00 and DMODE ≠ 00)

Writes to the alternate byte register (QUADSPI\_ABR) never trigger the communication start. If alternate bytes are required, they must be programmed before.

As soon as a command is started, BUSY (bit 5 of QUADSPI\_SR) is automatically set.

### FIFO and data management

In indirect mode, data go through a 32-byte FIFO which is internal to the QUADSPI. FLEVEL[5:0] (QUADSPI\_SR[13:8]) indicates how many bytes are currently being held in the FIFO.

In indirect-write mode (FMODE = 00), the firmware adds data to the FIFO when it writes QUADSPI\_DR. Word writes add 4 bytes to the FIFO, halfword writes add 2 bytes, and byte writes add only 1 byte. If the firmware adds too many bytes to the FIFO (more than is indicated by DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

Byte/halfword accesses to QUADSPI\_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES[4:0] is used to define a FIFO threshold. When the threshold is reached, FTF (FIFO threshold flag) is set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there are data in the FIFO after the last byte is read from the flash memory, regardless of the FTHRES setting.

In indirect-write mode, FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when FTF is set. If DMAEN = 1, a DMA transfer is initiated when FTF is set. FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data is transferred by the CPU or DMA).

### 15.3.6 QUADSPI automatic status-polling mode

In automatic status-polling mode, the QUADSPI periodically starts a command to read a defined number of status bytes (up to 4). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value.

Accesses to the flash memory begin in the same way as in indirect-read mode: if no address is required (AMODE = 00), accesses begin as soon as the QUADSPI\_CCR is written.

Otherwise, if an address is required, the first access begins when QUADSPI\_AR is written. BUSY goes high at this point and stays high even between the periodic accesses.

The contents of MASK[31:0] (QUADSPI\_PSMAR) are used to mask the data from the flash memory in automatic status-polling mode. If MASK[n] = 0, then bit n of the result is masked and not considered. If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] (QUADSPI\_PSMAR), then there is a match for bit n.

If the polling match mode bit (PMM, bit 23 of QUADSPI\_CR) is 0, then “AND” match mode is activated. This means status match flag (SMF) is set only when there is a match on all of the unmasked bits.

If PMM = 1, then “OR” match mode is activated. This means SMF is set if there is a match on any of the unmasked bits.

An interrupt is called when SMF is set if SMIE = 1.

If the automatic status-polling mode stop (APMS) bit is set, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1, and the periodic accesses continue until there is an abort or the QUADSPI is disabled (EN = 0).

The data register (QUADSPI\_DR) contains the latest received status bytes (the FIFO is deactivated). The content of the data register is not affected by the masking used in the matching logic. The FTF status bit is set as soon as a new reading of the status is complete, and FTF is cleared as soon as the data is read.

### 15.3.7 QUADSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

It is forbidden to access the quad-SPI flash bank area before having properly configured and enabled the QUADSPI peripheral.

No more than 256 Mbytes can be addressed even if the flash memory capacity is larger.

If an access is made to an address outside of the range defined by FSIZE but still within the 256-Mbyte range, then a bus error is given. The effect of this error depends on the bus master that attempted the access:

- If it is the Cortex CPU, bus fault exception is generated when enabled (or a HardFault exception when bus fault is disabled).
- If it is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

Byte, halfword, and word access types are all supported.

Support for execute in place (XIP) operation is implemented, where the QUADSPI anticipates the next access and load in advance the byte at the following address. If the subsequent access is indeed made at a continuous address, the access is completed faster since the value is already prefetched.

By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with NCS maintained low, even if no access to the flash memory occurs for a long time. Since flash memories tend to consume more when NCS is held low, the application may want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI\_CR) so that NCS is released after a period of TIMEOUT[15:0] (QUADSPI\_LPTR) cycles have elapsed without any access since when the FIFO becomes full with prefetch data.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is a timeout, there is an abort, or the peripheral is disabled.

### 15.3.8 QUADSPI flash memory configuration

The device configuration register (QUADSPI\_DCR) can be used to specify the characteristics of the external SPI flash memory.

The FSIZE[4:0] field defines the size of external memory using the following formula:

$$\text{Number of bytes in flash memory} = 2^{\text{FSIZE}+1}$$

FSIZE+1 is effectively the number of address bits required to address the flash memory. The flash memory capacity can be up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

When the QUADSPI executes two commands, one immediately after the other, it raises NCS high between the two commands for only one CLK cycle by default. If the flash memory requires more time between commands, the CSHT field can be used to specify the minimum number of CLK cycles (up to 8) that NCS must remain high.

The clock mode (CKMODE) bit indicates the CLK signal logic level in between commands (when NCS = 1).

### 15.3.9 QUADSPI delayed data sampling

By default, the QUADSPI samples the data driven by the flash memory one half of a CLK cycle after the flash memory drives the signal.

In case of external signal delays, it may be beneficial to sample the data later.

Using SSHIFT (bit 4 of QUADSPI\_CR), the sampling of the data can be shifted by half of a CLK cycle.

Clock shifting is not supported in DDR mode: SSHIFT must be clear when DDRM bit is set.

### 15.3.10 QUADSPI configuration

The QUADSPI configuration is done in two phases:

1. QUADSPI peripheral configuration
2. QUADSPI flash memory configuration

Once configured and enabled, the QUADSPI can be used in one of its three operating modes: indirect, automatic status-polling, or memory-mapped mode.

#### QUADSPI configuration

The QUADSPI is configured using QUADSPI\_CR. The user must configure the clock prescaler division factor and the sample shifting settings for the incoming data.

The DDR mode can be set through the DDRM bit. When setting the quad-SPI interface in DDR mode, the internal divider of kernel clock must be set with a division ratio of two or more. Once enabled, the address and the alternate bytes are sent on both clock edges, and the data are sent/received on both clock edges. Regardless of the DDRM bit setting, instructions are always sent in SDR mode.

DMA requests are enabled setting the DMAEN bit. In case of interrupt usage, their respective enable bit can be also set during this phase.

The FIFO level for either DMA request generation or interrupt generation is programmed in the FTHRES bits.

If a timeout counter is needed, the TCEN bit can be set and the timeout value programmed in the QUADSPI\_LPTR register.

### QUADSPI flash memory configuration

The parameters related to the targeted external flash memory are configured through the QUADSPI\_DCR register. The user must program the flash memory size in the FSIZE bits, the chip-select minimum high time in CSHT bits, and the functional mode (Mode 0 or Mode 3) in the MODE bit.

## 15.3.11 QUADSPI use

The operating mode is selected using FMODE[1:0] (QUADSPI\_CCR[27:26]).

### Indirect mode

When FMODE is programmed to 00, the indirect-write mode is selected and data can be sent to the flash memory. With FMODE = 01, the indirect-read mode is selected where data can be read from the flash memory.

When the QUADSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in QUADSPI\_DLR.
2. Specify the frame format, mode and instruction code in QUADSPI\_CCR.
3. Specify optional alternate byte to be sent right after the address phase in QUADSPI\_ABR.
4. Specify the operating mode in QUADSPI\_CR. If FMODE = 00 (indirect-write mode) and DMAEN = 1, then QUADSPI\_AR must be specified before QUADSPI\_CR. Otherwise QUADSPI\_DR can be written by the DMA before QUADSPI\_AR is updated (if the DMA controller has already been enabled).
5. Specify the targeted address in QUADSPI\_AR.
6. Read/write the data from/to the FIFO through QUADSPI\_DR.

When writing QUADSPI\_CR, the user specifies the following settings:

- enable bit (EN) set to 1
- DMA enable bit (DMAEN) for transferring data to/from RAM
- timeout counter enable bit (TCEN)
- sample shift setting (SSSHIFT)
- FIFO threshold level (FTRHES) to indicate when the FTF flag must be set
- interrupt enables
- automatic status-polling mode parameters: match mode and stop mode (valid when FMODE = 11)
- clock prescaler

When writing QUADSPI\_CCR, the user specifies the following parameters:

- instruction byte through INSTRUCTION bits
- the way the instruction has to be sent through the IMODE bits (1/2/4 lines)
- the way the address has to be sent through the ADMODE bits (None/1/2/4 lines)

- address size (8/16/24/32-bit) through ADSIZE bits
- the way the alternate bytes have to be sent through the ABMODE (None/1/2/4 lines)
- alternate bytes number (1/2/3/4) through the ABSIZE bits
- presence or not of dummy bytes through the DBMODE bit
- number of dummy bytes through the DCYC bits
- the way data have to be sent/received (none/1/2/4 lines) through DMODE bits

If neither QUADSPI\_AR nor QUADSPI\_DR need to be updated for a particular command, then the command sequence starts as soon as QUADSPI\_CCR is written. This is the case when both ADMODE and DMODE are 00, or if just ADMODE = 00 when in indirect read mode (FMODE = 01).

When an address is required (ADMODE is not 00) and the data register does not need to be written (when FMODE = 01 or DMODE = 00), the command sequence starts as soon as the address is updated with a write to QUADSPI\_AR.

In case of data transmission (FMODE = 00 and DMODE! = 00), the communication start is triggered by a write in the FIFO through QUADSPI\_DR.

### Automatic status-polling mode

This mode is enabled setting the FMODE field (QUADSPI\_CCR[27:26]) to 10. In this mode, the programmed frame is sent and the data retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in QUADSPI\_DLR, it is ignored and only 4 bytes are read.

The periodicity is specified in the QUADSPI\_PISR register.

Once the status data is retrieved, it can internally be processed:

- to set the status match flag and generate an interrupt if enabled
- to stop automatically the periodic retrieving of the status bytes

The received value can be masked with the value stored in QUADSPI\_PSMKR and ORed or ANDed with the value stored in QUADSPI\_PSMAR.

In case of match, the status match flag is set and an interrupt is generated if enabled, and the QUADSPI can be automatically stopped if the AMPS bit is set.

In any case, the latest retrieved value is available in QUADSPI\_DR.

### Memory-mapped mode

In memory-mapped mode, the external flash memory is seen as an internal memory but with some latency during accesses. Only read operations are allowed to the external flash memory in this mode.

The memory-mapped mode is entered by setting FMODE to 11 in QUADSPI\_CCR.

The programmed instruction and frame is sent when a master is accessing the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate linear reads. Any access to QUADSPI\_DR in this mode returns zero.

QUADSPI\_DLR has no meaning in memory-mapped mode.

### 15.3.12 Sending the instruction only once

Some flash memories (for example: Winbound) provide a mode where an instruction must be sent only with the first command sequence, while subsequent commands start directly with the address. One can take advantage of such a feature using the SIOO bit (QUADSPI\_CCR[28]).

SIOO is valid for all functional modes (indirect, automatic status-polling, and memory-mapped). If the SIOO bit is set, the instruction is sent only for the first command following a write to QUADSPI\_CCR. Subsequent command sequences skip the instruction phase, until there is a write to QUADSPI\_CCR.

SIOO has no effect when IMODE = 00 (no instruction).

### 15.3.13 QUADSPI error management

An error can be generated in the following case:

- In indirect mode or automatic status-polling mode when a wrong address is programmed in QUADSPI\_AR (according to the flash memory size defined by FSIZE[4:0] in the QUADSPI\_DCR), TEF is set and an interrupt is generated if enabled. Also in indirect mode, if the address plus the data length exceeds the flash memory size, TEF is set as soon as the access is triggered.
- In memory-mapped mode, when an out-of-range access is done by a master or when the QUADSPI is disabled, a bus error is generated as a response to the faulty bus master request.

When a master is accessing the memory mapped space while the memory-mapped mode is disabled, a bus error is generated as a response to the faulty bus master request.

### 15.3.14 QUADSPI busy bit and abort functionality

Once the QUADSPI starts an operation with the flash memory, the BUSY bit is automatically set in QUADSPI\_SR.

In indirect mode, BUSY is reset once the QUADSPI has completed the requested command sequence, and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1, or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on a timeout event or on an abort.

Any operation can be aborted by setting the ABORT bit in QUADSPI\_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

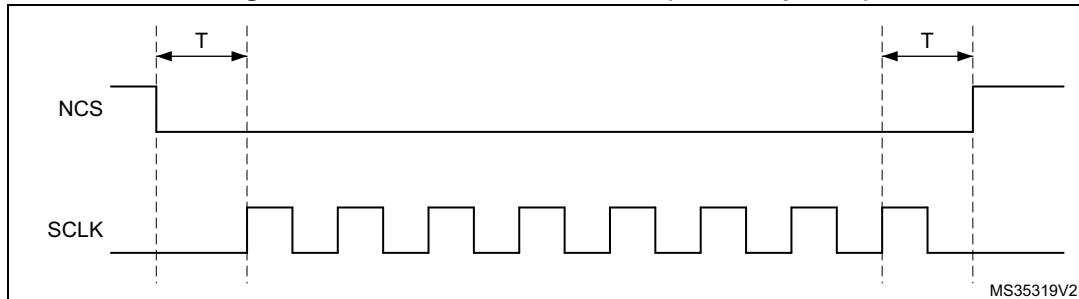
*Note:* Some flash memories may misbehave if a write operation to a status registers is aborted.

### 15.3.15 NCS behavior

By default, NCS is high, deselecting the external flash memory. NCS falls before an operation begins and rises as soon as it finishes.

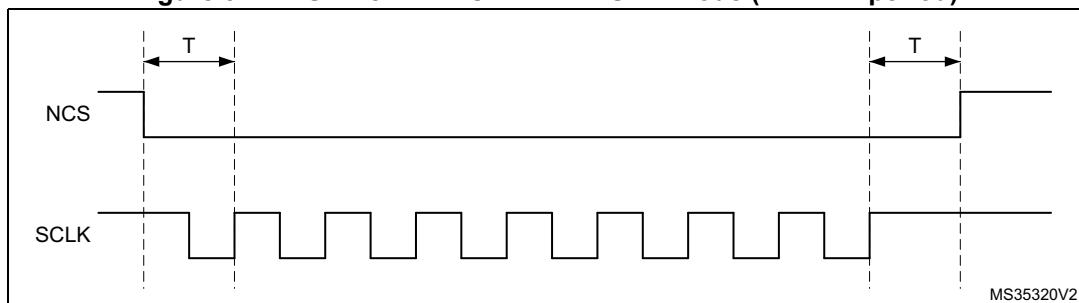
When CKMODE = 0 ("mode0", where CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge, as shown in the figure below.

Figure 36. NCS when CKMODE = 0 (T = CLK period)



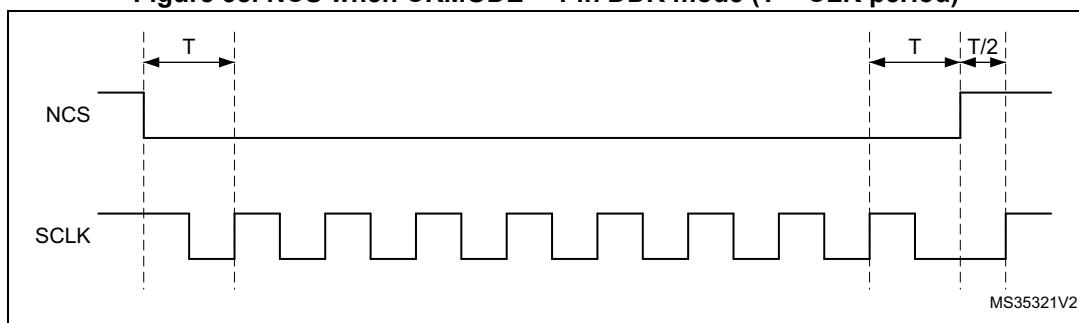
When CKMODE=1 ("mode3", where CLK goes high when no operation is in progress) and DDRM=0 (SDR mode), NCS still falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge, as shown in the figure below.

Figure 37. NCS when CKMODE = 1 in SDR mode (T = CLK period)



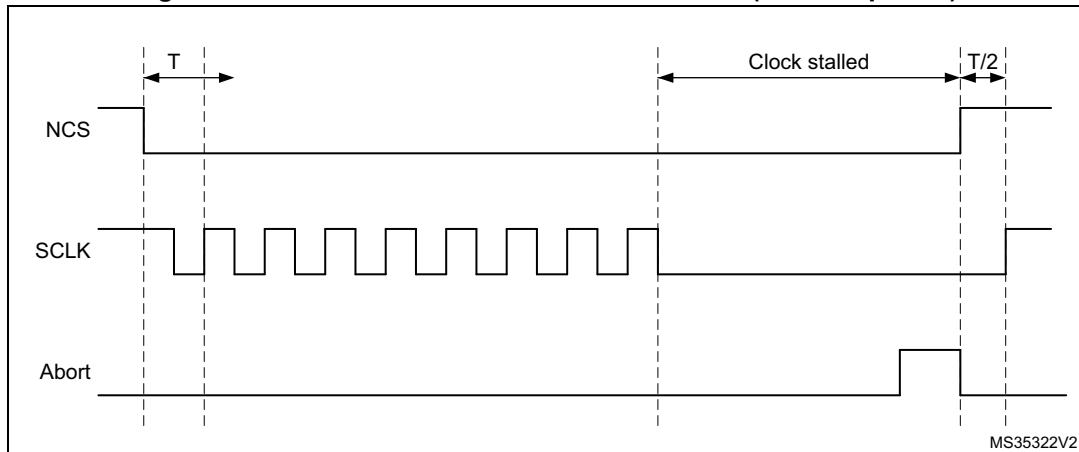
When CKMODE = 1 ("mode3") and DDRM = 1 (DDR mode), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final active rising CLK edge, as shown in the figure below. Because DDR operations must finish with a falling edge, CLK is low when NCS rises, and CLK rises back up one half of a CLK cycle afterwards.

Figure 38. NCS when CKMODE = 1 in DDR mode (T = CLK period)



When the FIFO stays full in a read operation or if the FIFO stays empty in a write operation, the operation stalls and CLK stays low until firmware services the FIFO. If an abort occurs when an operation is stalled, NCS rises just after the abort is requested and then CLK rises one half of a CLK cycle later, as shown in [Figure 39](#).

**Figure 39. NCS when CKMODE = 1 with an abort (T = CLK period)**



## 15.4 QUADSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

**Table 72. QUADSPI interrupt requests**

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

## 15.5 QUADSPI registers

### 15.5.1 QUADSPI control register (QUADSPI\_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
PRESCALER[7:0]										PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Res.	Res.	Res.	FTHRES[4:0]				Res.	Res.	Res.	SSHIFT	TCEN	DMAEN	ABORT	EN			
			rw	rw	rw	rw				rw	rw	rw	rw	rw			

Bits 31:24 **PRESCALER[7:0]**: Clock prescaler

This field defines the scaler factor for generating CLK based on the clock (value+1).

0:  $F_{CLK} = F$ , clock used directly as QUADSPI CLK (prescaler bypassed)

1:  $F_{CLK} = F/2$

2:  $F_{CLK} = F/3$

...

255:  $F_{CLK} = F/256$

For odd clock division factors, CLK duty cycle is not 50%. The clock signal remains low one cycle longer than it stays high.

When setting quad-SPI interface in DDR mode, the prescaler must be set with a division ratio of two or more.

*Note: This field can be modified only when BUSY = 0.*

Bit 23 **PMM**: Polling match mode

This bit indicates which method must be used for determining a “match” during automatic status-polling mode.

0: AND match mode. SMF is set if all the unmasked bits received from the flash memory match the corresponding bits in the match register.

1: OR match mode. SMF is set if any one of the unmasked bits received from the flash memory matches its corresponding bit in the match register.

*Note: This bit can be modified only when BUSY = 0.*

Bit 22 **APMS**: Automatic status-polling mode stop

This bit determines if automatic status-polling is stopped after a match.

0: Automatic status-polling mode is stopped only by abort or by disabling the QUADSPI.

1: Automatic status-polling mode stops as soon as there is a match.

*Note: This bit can be modified only when BUSY = 0.*

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TOIE**: Timeout interrupt enable

This bit enables the timeout interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bit 19 **SMIE**: Status match interrupt enable  
 This bit enables the status match interrupt.  
 0: Interrupt disabled  
 1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable  
 This bit enables the FIFO threshold interrupt.  
 0: Interrupt disabled  
 1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable  
 This bit enables the transfer complete interrupt.  
 0: Interrupt disabled  
 1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable  
 This bit enables the transfer error interrupt.  
 0: Interrupt disabled  
 1: Interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **FTHRES[4:0]**: FIFO threshold level  
 This field defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag (bit FTF in register QUADSPI\_SR) to be set.  
 0: In indirect-write mode (FMODE = 00), FTF is set if there are one or more free bytes location left in the FIFO, or indirect-read mode (FMODE = 01), FTF is set if there are one or more valid bytes that can be read from the FIFO.  
 1: In indirect-write mode (FMODE = 00), FTF is set if there are two or more free bytes location left in the FIFO, or indirect-read mode (FMODE = 01), FTF is set if there are two or more valid bytes that can be read from the FIFO  
 ...  
 31: In indirect-write mode (FMODE = 00), FTF is set if there are 32 free bytes location left in the FIFO, or indirect-read mode (FMODE = 01), FTF is set if there are 32 valid bytes that can be read from the FIFO.  
 If DMAEN = 1, then the DMA controller for the corresponding channel must be disabled before changing the FTHRES value.

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

Bit 4 **SSHIFT**: Sample shift  
 By default, the QUADSPI samples data 1/2 of a CLK cycle after the data is driven by the flash memory. This bit allows the data to be sampled later in order to account for external signal delays.  
 0: No shift  
 1: 1/2 cycle shift  
 The firmware must assure that SSHIFT = 0 when in DDR mode (when DDRM = 1).  
*Note: This field can be modified only when BUSY = 0.*

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when memory-mapped mode (FMODE = 11) is selected. Activating this bit causes the NCS to be released (and thus reduces consumption) if there has not been an access after a certain amount of time, where this time is defined by TIMEOUT[15:0] (QUADSPI\_LPTR). This bit enables the timeout counter.

By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with NCS maintained low, even if no access to the flash memory occurs for a long time. Since flash memories tend to consume more when NCS is held low, the application may want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI\_CR) so that NCS is released after a period of TIMEOUT[15:0] (QUADSPI\_LPTR) cycles have elapsed without an access since when the FIFO becomes full with prefetch data.

0: Timeout counter is disabled, and thus the NCS remains active indefinitely after an access in memory-mapped mode.

1: Timeout counter is enabled, and thus the NCS is released in memory-mapped mode after TIMEOUT[15:0] cycles of flash memory inactivity.

*Note: This bit can be modified only when BUSY = 0.*

Bit 2 **DMAEN**: DMA enable

In indirect mode, the DMA can be used to input or output data via the QUADSPI\_DR register. DMA transfers are initiated when the FIFO threshold flag, FTF, is set.

0: DMA is disabled for indirect mode.

1: DMA is enabled for indirect mode.

Bit 1 **ABORT**: Abort request

This bit aborts the ongoing command sequence. It is automatically reset once the abort is complete. This bit stops the current transfer.

In automatic status-polling or memory-mapped mode, this bit also reset APM or DM bit.

0: No abort requested

1: Abort requested

Bit 0 **EN**: QUADSPI enable

0: QUADSPI disabled

1: QUADSPI enabled

## 15.5.2 QUADSPI device configuration register (QUADSPI\_DCR)

Address offset: 0x004

Reset value: 0x0000 0000

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

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

Bits 20:16 **FSIZE[4:0]**: Flash memory size

This field defines the size of external memory using the following formula:

$$\text{Number of bytes in flash memory} = 2^{\lceil \text{FSIZE} + 1 \rceil}$$

FSIZE+1 is effectively the number of address bits required to address the flash memory.

The flash memory capacity can be up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

*Note: This field can be modified only when BUSY = 0.*

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **CSHT[2:0]**: Chip select high time

CSHT+1 defines the minimum number of CLK cycles which the chip select (NCS) must remain high between commands issued to the flash memory.

0: NCS stays high for at least 1 cycle between flash memory commands

1: NCS stays high for at least 2 cycles between flash memory commands

...

7: NCS stays high for at least 8 cycles between flash memory commands

*Note: This field can be modified only when BUSY = 0.*

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **CKMODE**: Mode 0/mode 3

This bit indicates the level that CLK takes between commands (when NCS = 1).

0: CLK must stay low while NCS is high (chip select released). This is referred to as mode 0.

1: CLK must stay high while NCS is high (chip select released). This is referred to as mode 3.

*Note: This field can be modified only when BUSY = 0.*

### 15.5.3 QUADSPI status register (QUADSPI\_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FLEVEL[5:0]						Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
		r	r	r	r	r	r			r	r	r	r	r	r

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

Bits 13:8 **FLEVEL[5:0]**: FIFO level

This field gives the number of valid bytes which are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full. In memory-mapped mode and in automatic status-polling mode, FLEVEL is zero.

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

Bit 5 **BUSY**: Busy

This bit is set when an operation is on going. This bit clears automatically when the operation with the flash memory is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 **SMF**: Status match flag

This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (QUADSPI\_PSMAR). It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold is reached, or if there is any data left in the FIFO after reads from the flash memory are complete. It is cleared automatically as soon as threshold condition is no longer true.

In automatic status-polling mode this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag

This bit is set in indirect mode when the programmed number of data is transferred or in any mode when the transfer is aborted. It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag

This bit is set in indirect mode when an invalid address is being accessed in indirect mode. It is cleared by writing 1 to CTEF.

#### 15.5.4 QUADSPI flag clear register (QUADSPI\_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CTOF	CSMF	Res.	CTCF	CTEF										
											w	w		w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CTOF**: Clear timeout flag

Writing 1 clears the TOF flag in QUADSPI\_SR.

Bit 3 **CSMF**: Clear status match flag

Writing 1 clears the SMF flag in QUADSPI\_SR.

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

Bit 1 **CTCF**: Clear transfer complete flag

Writing 1 clears the TCF flag in QUADSPI\_SR.

Bit 0 **CTEF**: Clear transfer error flag

Writing 1 clears the TEF flag in QUADSPI\_SR.

### 15.5.5 QUADSPI data length register (QUADSPI\_DLR)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **DL[31:0]**: Data length

Number of data to be retrieved (value+1) in indirect and automatic status-polling modes. A value no greater than 3 (indicating 4 bytes) must be used for automatic status-polling mode.

All 1s in indirect mode means undefined length, where the QUADSPI continues until the end of memory, as defined by FSIZE.

0x0000\_0000: 1 byte is to be transferred

0x0000\_0001: 2 bytes are to be transferred

0x0000\_0002: 3 bytes are to be transferred

0x0000\_0003: 4 bytes are to be transferred

...

0xFFFF\_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred

0xFFFF\_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred

0xFFFF\_FFFF: undefined length -- all bytes until the end of flash memory (as defined by FSIZE) are to be transferred. Continue reading indefinitely if FSIZE = 0x1F.

This field has no effect when in memory-mapped mode (FMODE = 10).

*Note: This field can be written only when BUSY = 0.*

### 15.5.6 QUADSPI communication configuration register (QUADSPI\_CCR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DDRM	Res.	Res.	SIO0	FMODE[1:0]		DMODE[1:0]		Res.	DCYC[4:0]					ABSIZE[1:0]	
rw			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABMODE[1:0]		ADSIZE[1:0]		ADMODE[1:0]		IMODE[1:0]		INSTRUCTION[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DDRM**: Double data rate mode

This bit sets the DDR mode for the address, alternate byte and data phase:

0: DDR mode disabled

1: DDR mode enabled

*Note: This field can be written only when BUSY = 0.*

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **SIOO**: Send instruction only once mode

This bit has no effect when IMODE = 00. See [Section 15.3.12](#) for more details.

0: Instruction sent on every transaction

1: Instruction sent only for the first command

*Note: This field can be written only when BUSY = 0.*

Bits 27:26 **FMODE[1:0]**: Functional mode

This field defines the QUADSPI functional mode of operation.

00: Indirect-write mode

01: Indirect-read mode

10: Automatic status-polling mode

11: Memory-mapped mode

If DMAEN = 1 already, the DMA controller for the corresponding channel must be disabled before changing the FMODE value.

*Note: This field can be written only when BUSY = 0.*

Bits 25:24 **DMODE[1:0]**: Data mode

This field defines the data phase mode of operation:

00: No data

01: Data on a single line

10: Data on two lines

11: Data on four lines

This field also determines the dummy phase mode of operation.

*Note: This field can be written only when BUSY = 0.*

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **DCYC[4:0]**: Number of dummy cycles

This field defines the duration of the dummy phase. In both SDR and DDR modes, it specifies a number of CLK cycles (0-31).

*Note: This field can be written only when BUSY = 0.*

Bits 17:16 **ABSIZE[1:0]**: Alternate-byte size

This bit defines the size of alternate bytes.

00: 8-bit alternate byte

01: 16-bit alternate bytes

10: 24-bit alternate bytes

11: 32-bit alternate bytes

*Note: This field can be written only when BUSY = 0.*

Bits 15:14 **ABMODE[1:0]**: Alternate byte mode

This field defines the alternate-byte phase mode of operation.

00: No alternate bytes

01: Alternate bytes on a single line

10: Alternate bytes on two lines

11: Alternate bytes on four lines

*Note: This field can be written only when BUSY = 0.*

Bits 13:12 **ADSIZE[1:0]**: Address size

This bit defines address size:

00: 8-bit address

01: 16-bit address

10: 24-bit address

11: 32-bit address

*Note: This field can be written only when BUSY = 0.*

Bits 11:10 **ADMODE[1:0]**: Address mode

This field defines the address phase mode of operation.

00: No address

01: Address on a single line

10: Address on two lines

11: Address on four lines

*Note: This field can be written only when BUSY = 0.*

Bits 9:8 **IMODE[1:0]**: Instruction mode

This field defines the instruction phase mode of operation.

00: No instruction

01: Instruction on a single line

10: Instruction on two lines

11: Instruction on four lines

*Note: This field can be written only when BUSY = 0.*

Bits 7:0 **INSTRUCTION[7:0]**: Instruction

Instruction to be sent to the external SPI device.

*Note: This field can be written only when BUSY = 0.*

**15.5.7 QUADSPI address register (QUADSPI\_AR)**

Address offset: 0x018

Reset value: 0x0000 0000

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

Bits 31:0 **ADDRESS[31:0]**: Address

This field contains the address to be sent to the external flash memory.

Writes to this field are ignored when BUSY = 1 or when FMODE = 11 (memory-mapped mode).

**15.5.8 QUADSPI alternate-byte register (QUADSPI\_ABR)**

Address offset: 0x01C

Reset value: 0x0000 0000

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

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be send to the external SPI device right after the address.

*Note: This field can be written only when BUSY = 0.*

### 15.5.9 QUADSPI data register (QUADSPI\_DR)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device.

In indirect write mode, data written to this register is stored on the FIFO before it is sent to the flash memory during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect read mode, reading this register gives (via the FIFO) the data which was received from the flash memory. If the FIFO does not have as many bytes as requested by the read operation and if BUSY=1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic status-polling mode, this register contains the last data read from the flash memory (without masking).

Word, halfword, and byte accesses to this register are supported. In indirect write mode, a byte write adds 1 byte to the FIFO, a halfword write 2, and a word write 4. Similarly, in indirect read mode, a byte read removes 1 byte from the FIFO, a halfword read 2, and a word read 4. Accesses in indirect mode must be aligned to the bottom of this register: a byte read must read DATA[7:0] and a halfword read must read DATA[15:0].

### 15.5.10 QUADSPI polling status mask register (QUADSPI\_PSMKR)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in automatic status-polling mode.

For bit n:

0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic

1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic

*Note: This field can be written only when BUSY = 0.*

**15.5.11 QUADSPI polling status match register (QUADSPI\_PSMAR)**

Address offset: 0x028

Reset value: 0x0000 0000

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

Bits 31:0 **MATCH[31:0]**: Status match

Value to be compared with the masked status register to get a match.

*Note: This field can be written only when BUSY = 0.*

**15.5.12 QUADSPI polling interval register (QUADSPI\_PIR)**

Address offset: 0x02C

Reset value: 0x0000 0000

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

Number of CLK cycles between two read during automatic status-polling phases.

*Note: This field can be written only when BUSY = 0.*

### 15.5.13 QUADSPI low-power timeout register (QUADSPI\_LPTR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
TIMEOUT[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 **TIMEOUT[15:0]**: Timeout period

After each access in memory-mapped mode, the QUADSPI prefetches the subsequent bytes and holds these bytes in the FIFO. This field indicates how many CLK cycles the QUADSPI waits after the FIFO becomes full until it raises NCS, putting the flash memory in a lower-consumption state.

*Note: This field can be written only when BUSY = 0.*

### 15.5.14 QUADSPI register map

Table 73. QUADSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x000	QUADSPI_CR	PRESCALER[7:0]							PMM	PMM	APMS	APMS	Res.	Res.	FTHRES[4:0]		
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	QUADSPI_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSHT	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	QUADSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[5:0]		
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	QUADSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTRUCTION[7:0]	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	QUADSPI_DLR	DL[31:0]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	QUADSPI_CCR	DDRM	Res.	Res.	Res.												
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	QUADSPI_AR	ADDRESS[31:0]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 73. QUADSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	QUADSPI_ABR																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x020	QUADSPI_DR																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x024	QUADSPI_PSMKR																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x028	QUADSPI_PSMAR																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x02C	QUADSPI_PIR	Res.																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x030	QUADSPI_LPTR	Res.																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

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

## 16 Analog-to-digital converter (ADC)

### 16.1 Introduction

The ADC consists of a 12-bit successive approximation analog-to-digital converter.

The ADC has up to 19 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The ADC is mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows to improve analog performance while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

### 16.2 ADC main features

- High-performance features
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time is independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Manage single-ended or differential inputs
  - AHB slave bus interface to allow fast data handling
  - Self-calibration
  - Channel-wise programmable sampling time
  - Up to four injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - 4 dedicated data registers for the injected channels
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256
  - Programmable data shift up to 8-bit
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency

- application (auto-delayed mode)
- Number of external analog input channels
  - Up to 5 fast channels from GPIO pads
  - Up to 11 slow channels from GPIO pads
- In addition, there are several internal dedicated channels
  - The internal reference voltage ( $V_{REFINT}$ ), connected to ADC1
  - The internal temperature sensor ( $V_{TS}$ ), connected to ADC1
  - The  $V_{BAT}$  monitoring channel ( $V_{BAT}/3$ ), connected to ADC1
- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - The ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs
- ADC input range:  $V_{REF-} \leq V_{IN} \leq V_{REF+}$

*Figure 40* shows the block diagram of one ADC.

### 16.3 ADC implementation

Table 74. ADC features

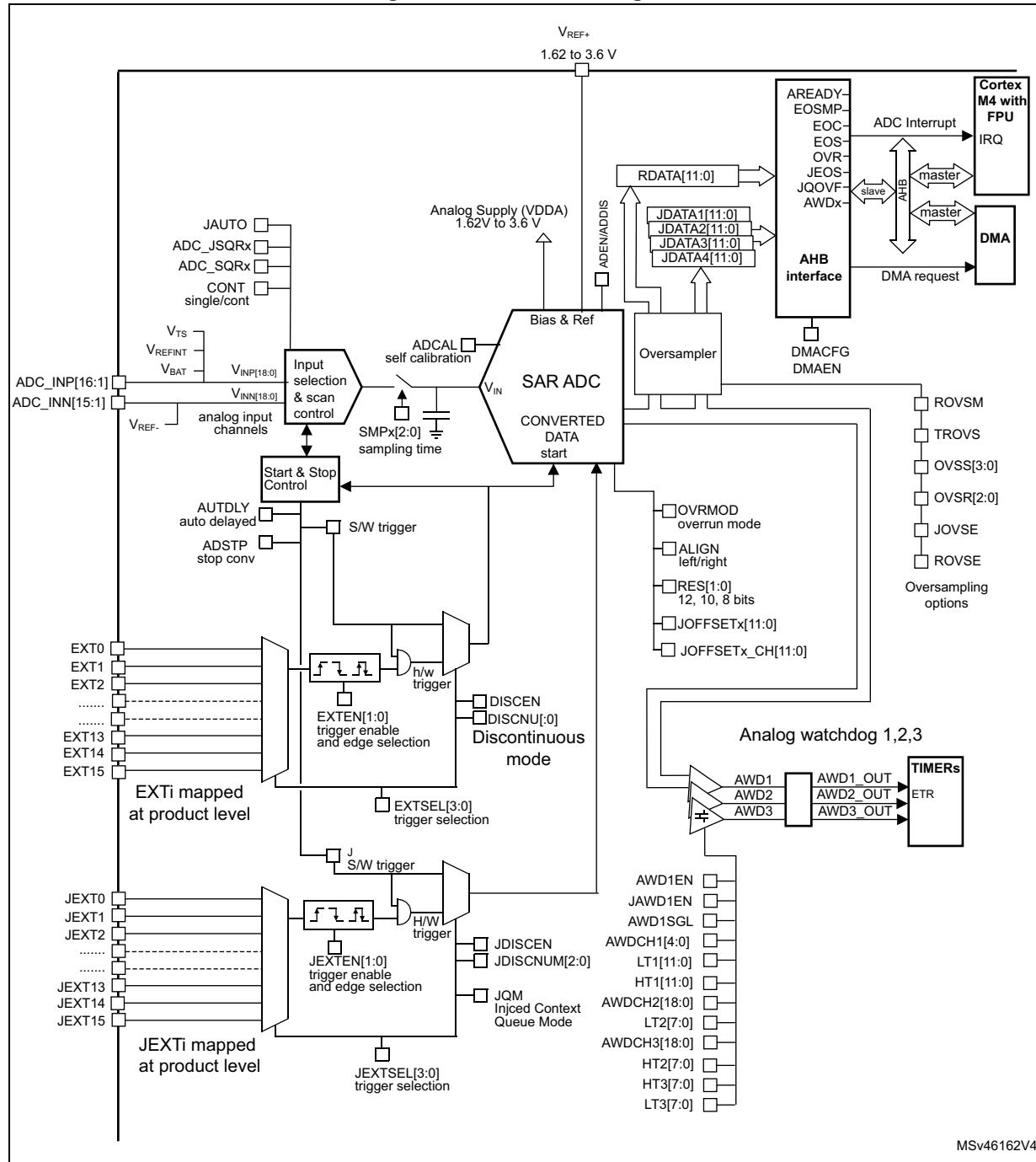
Feature	STM32WB55xx	STM32WB35xx
ADC1_INN	1 to 15	5 to 12 and 15
ADC1_INP	1 to 16	5 to 13, 15 to 16

## 16.4 ADC functional description

### 16.4.1 ADC block diagram

Figure 40 shows the ADC block diagram and Table 76 gives the ADC pin description.

Figure 40. ADC block diagram



### 16.4.2 ADC pins and internal signals

Table 75. ADC internal input/output signals

Internal signal name	Signal type	Description
EXT[15:0]	Inputs	Up to 16 external trigger inputs for the regular conversions (can be connected to on-chip timers).
JEXT[15:0]	Inputs	Up to 16 external trigger inputs for the injected conversions (can be connected to on-chip timers).
ADC_AWD <sub>x</sub> _OUT	Output	Internal analog watchdog output signal connected to on-chip timers (x = Analog watchdog number 1,2,3)
V <sub>TS</sub>	Input	Output voltage from internal temperature sensor
V <sub>REFINT</sub>	Input	Output voltage from internal reference voltage
V <sub>BAT</sub>	Input supply	External battery voltage supply

Table 76. ADC input/output pins

Pin name	Signal type	Comments
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC
VDDA	Input, analog supply	Analog power supply equal V <sub>DDA</sub>
VREF-	Input, analog reference negative	The lower/negative reference voltage for the ADC. V <sub>REF-</sub> is internally connected to V <sub>SSA</sub>
VSSA	Input, analog supply ground	Ground for analog power supply. On device package which do not have a dedicated V <sub>SSA</sub> pin, V <sub>SSA</sub> is internally connected to V <sub>SS</sub> .
V <sub>INPi</sub>	Positive analog input channels	Connected either to ADC1_INPi external channels or to internal channels. This input is converted in single-ended mode
V <sub>INNi</sub>	Negative analog input channels	Connected either to V <sub>REF-</sub> or to external channels: ADC1_INNi and ADC1_INP[i+1].
ADC1_INNi	Negative external analog input signals	Up to 15 analog input channels Refer to <a href="#">Section 16.4.4: ADC1 connectivity</a> for details.
ADC1_INPi	Positive external analog input signals	Up to 16 analog input channels Refer to <a href="#">Section 16.4.4: ADC1 connectivity</a> for details

### 16.4.3 ADC clocks

#### Dual clock domain architecture

The dual clock-domain architecture means that the ADC clock is independent from the AHB bus clock.

The input clock is and can be selected between two different clock sources (see [Figure 41: ADC clock scheme](#)):

1. The ADC clock can be a specific clock source, derived from the following clock sources:
  - The system clock
  - PLLSAI1 (single ADC implementation)Refer to RCC Section for more information on how to generate ADC dedicated clock. To select this scheme, bits CKMODE[1:0] of the ADCx\_CCR register must be reset.
2. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). In this mode, a programmable divider factor can be selected (/1, 2 or 4 according to bits CKMODE[1:0]).

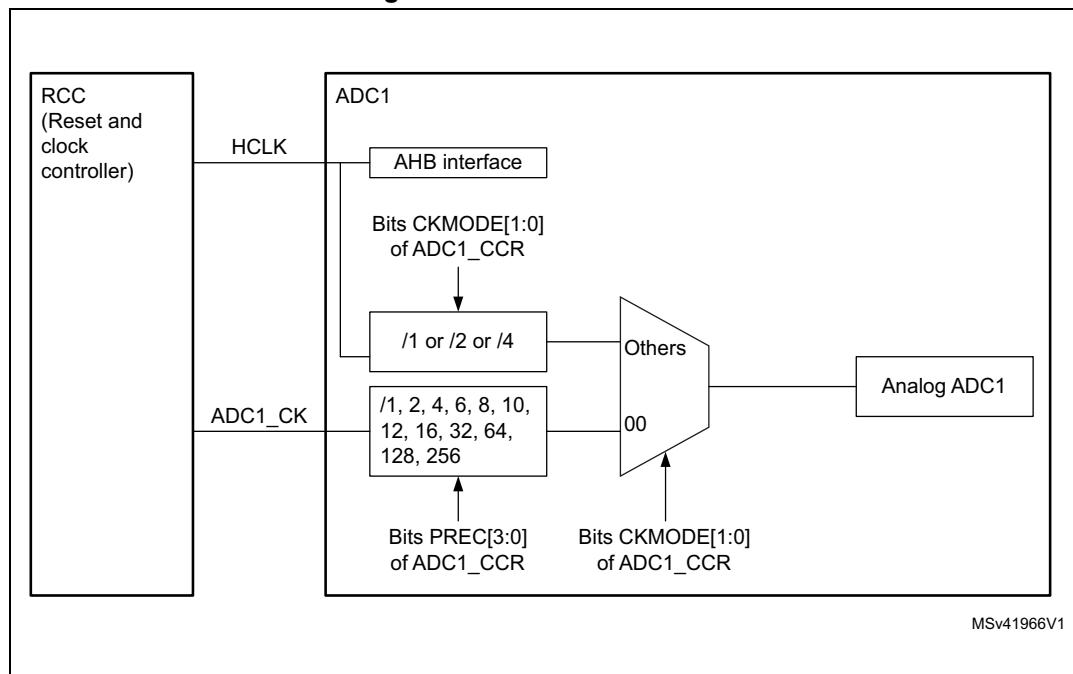
To select this scheme, bits CKMODE[1:0] of the ADCx\_CCR register must be different from 00.

**Note:** *For option 2), a prescaling factor of 1 (CKMODE[1:0] = 01) can be used only if the AHB prescaler is set (HPRE[3:0] = 0xxx in RCC\_CFGR register).*

Option 1) has the advantage of reaching the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, 256; using the prescaler configured with bits PRESC[3:0] in the ADCx\_CCR register.

Option 2) has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant time is added by the resynchronizations between the two clock domains).

Figure 41. ADC clock scheme



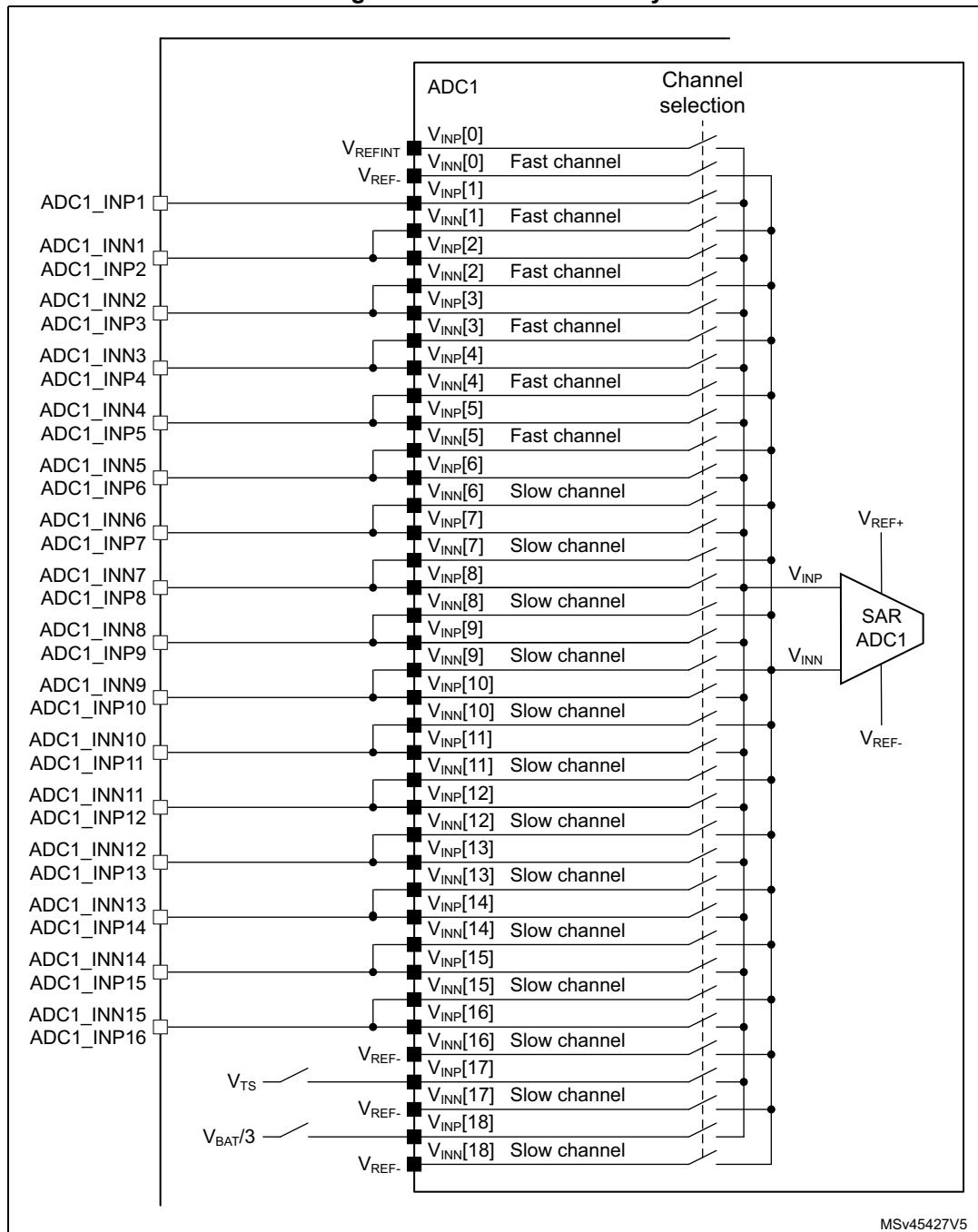
### Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{HCLK} \geq F_{ADC} / 4$  if the resolution of all channels are 12-bit or 10-bit
- $F_{HCLK} \geq F_{ADC} / 3$  if there are some channels with resolutions equal to 8-bit (and none with lower resolution)
- $F_{HCLK} \geq F_{ADC} / 2$  if there are some channels with resolutions equal to 6-bit

### 16.4.4 ADC1 connectivity

Figure 42. ADC1 connectivity



### 16.4.5 Slave AHB interface

The ADC implements an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

### 16.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in Deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPPWD is 1 in the ADC\_CR register).

To start ADC operations, follow the sequence below:

1. Exit Deep-power-down mode by clearing DEEPPWD bit.
2. Enable the ADC voltage regulator by setting ADVREGEN.
3. Wait for the startup time to configure the ADC (refer to the device datasheet for the value of the startup time).

When ADC operations are complete, the ADC can be disabled (ADEN = 0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN = 0.

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC Deep-power-down mode by setting bit DEEPPWD = 1 into ADC\_CR register. This is particularly interesting before entering Stop mode.

*Note: Writing DEEPPWD = 1 automatically disables the ADC voltage regulator and bit ADVREGEN is automatically cleared.*

*When the internal voltage regulator is disabled (ADVREGEN = 0), the internal analog calibration is kept.*

In ADC Deep-power-down mode (DEEPPWD = 1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or re-apply the calibration factor which was previously saved (refer to [Section 16.4.8: Calibration \(ADCAL, ADCALDIF, ADC\\_CALFACT\)](#)).

Before entering Stop 2 mode, the ADC needs to be disabled by setting ADDIS and clearing ADVREGEN.

### 16.4.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by programming DIFSEL[i] bits in the ADC\_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN = 0). Note that the DIFSEL[i] bits corresponding to single-ended channels are always programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the ADCy\_INPx external voltage equal to  $V_{INP[i]}$  (positive input) and  $V_{REF-}$  (negative input).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the ADCy\_INPx external voltage positive input equal to  $V_{INP[i]}$ , and the ADCy\_INN<sub>x</sub> negative input equal to  $V_{INN[i]}$ .

The input voltage in differential mode ranges from  $V_{REF-}$  to  $V_{REF+}$ , which makes a full scale range of  $2 \times V_{REF+}$ . When  $V_{INP[i]}$  equals  $V_{REF-}$ ,  $V_{INN[i]}$  equals  $V_{REF+}$  and the maximum negative input differential voltage ( $V_{REF-}$ ) corresponds to 0x000 ADC output. When  $V_{INP[i]}$  equals  $V_{REF+}$ ,  $V_{INN[i]}$  equals  $V_{REF-}$  and the maximum positive input differential voltage ( $V_{REF+}$ ) corresponds to 0xFFFF ADC output. When  $V_{INP[i]}$  and  $V_{INN[i]}$  are connected together, the zero input differential voltage corresponds to 0x800 ADC output.

The ADC sensitivity in differential mode is twice smaller than in single-ended mode.

When ADC is configured as differential mode, both inputs should be biased at  $(V_{REF+}) / 2$  voltage. Refer to the device datasheet for the allowed common mode input voltage  $V_{CMIN}$ .

The input signals are supposed to be differential (common mode voltage should be fixed).

Internal channels (such as  $V_{TS}$  and  $V_{REFINT}$ ) are used in single-ended mode only.

For a complete description of how the input channels are connected, refer to [Section 16.4.4: ADC1 connectivity](#).

**Caution:** When configuring the channel “i” in differential input mode, its negative input voltage  $V_{INN[i]}$  is connected to another channel. As a consequence, this channel is no longer usable in single-ended mode or in differential mode and must never be configured to be converted.

### 16.4.8 Calibration (ADCAL, ADCALDIF, ADC\_CALFACT)

The ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 7-bit wide and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF = 0 before launching a calibration to be applied for single-ended input conversions.
- Write ADCALDIF = 1 before launching a calibration to be applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL = 1. Calibration can only be initiated when the ADC is disabled (when ADEN = 0). ADCAL bit stays at 1 during all the

calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT\_S[6:0] or CALFACT\_D[6:0] of ADC\_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

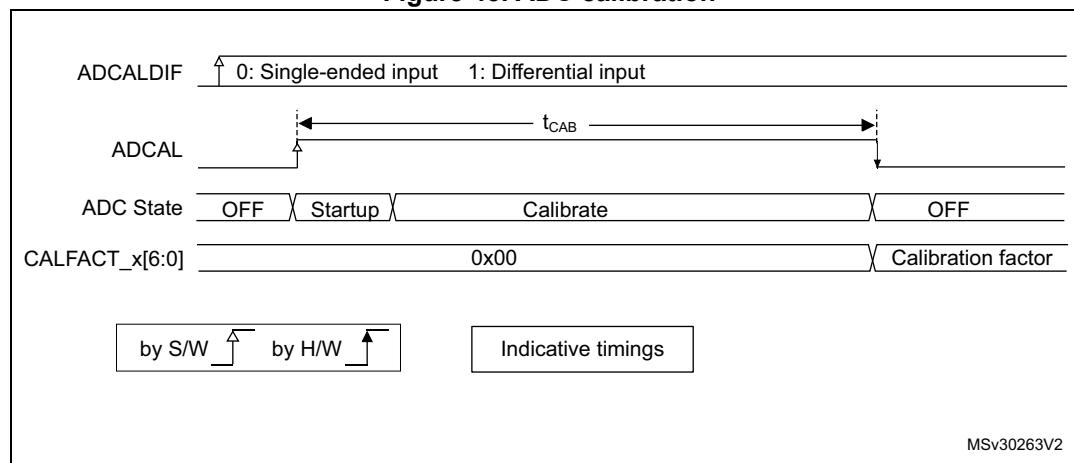
The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in Standby or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC\_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0 and JADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when  $V_{REF+}$  voltage changed more than 10%.

### Software procedure to calibrate the ADC

1. Ensure DEEPPWD = 0, ADVREGEN = 1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN = 0.
3. Select the input mode for this calibration by setting ADCALDIF = 0 (single-ended input) or ADCALDIF = 1 (differential input).
4. Set ADCAL.
5. Wait until ADCAL = 0.
6. The calibration factor can be read from ADC\_CALFACT register.

**Figure 43. ADC calibration**

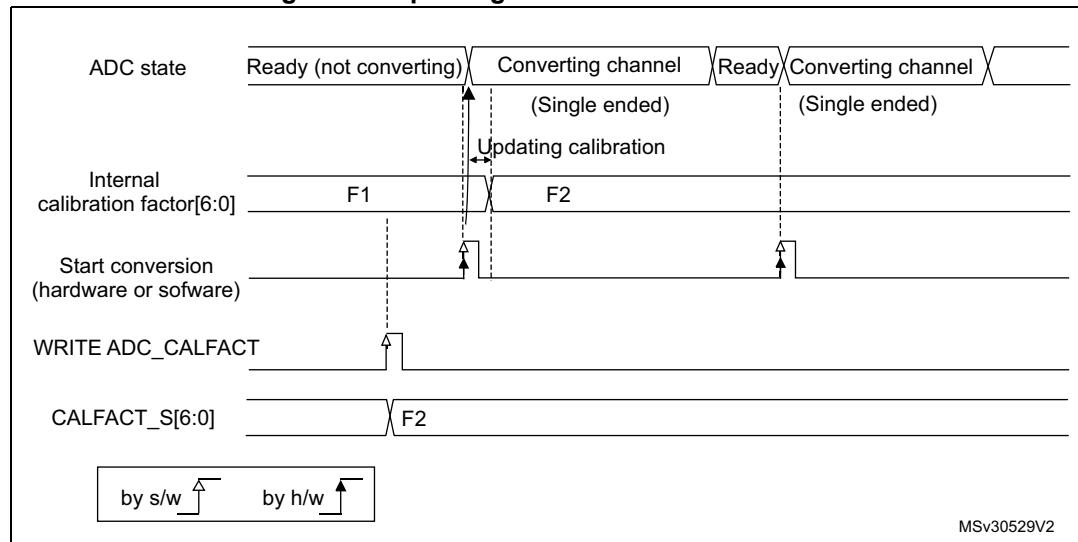


MSv30263V2

### Software procedure to re-inject a calibration factor into the ADC

1. Ensure ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT\_S and CALFACT\_D with the new calibration factors.
3. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT\_S for single-ended input channel or bits CALFACT\_D for differential input channel.

Figure 44. Updating the ADC calibration factor

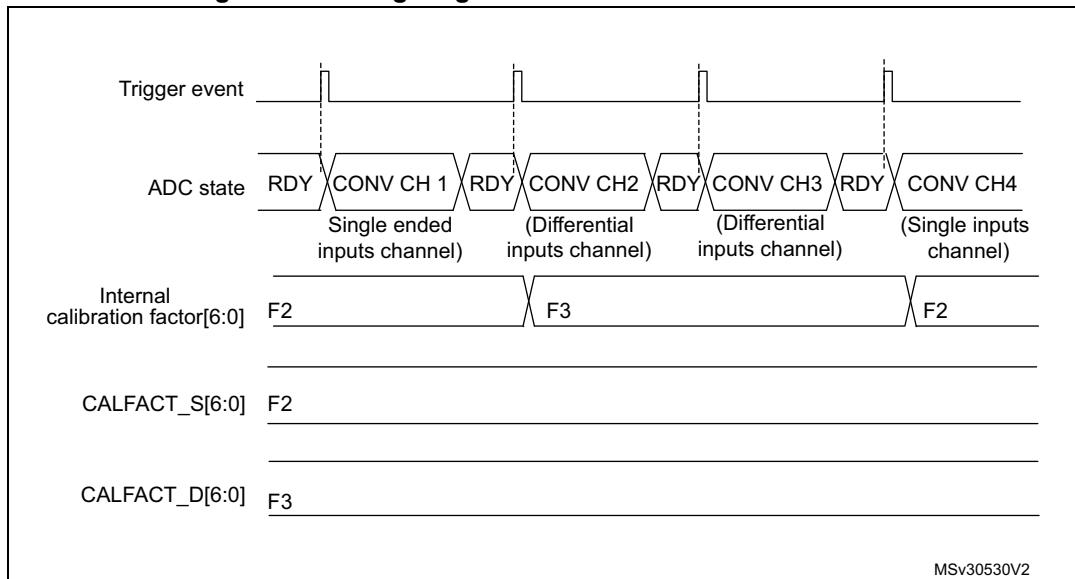


### Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF = 0). This updates the register CALFACT\_S[6:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF = 1). This updates the register CALFACT\_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration is automatically injected into the analog ADC.

Figure 45. Mixing single-ended and differential channels



#### 16.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in [Section 16.4.6: ADC Deep-power-down mode \(DEEPPWD\) and ADC voltage regulator \(ADVREGEN\)](#).

Once DEEPPWD = 0 and ADVREGEN = 1, the ADC can be enabled and the ADC needs a stabilization time of  $t_{STAB}$  before it starts converting accurately, as shown in [Figure 46](#). Two control bits enable or disable the ADC:

- ADEN = 1 enables the ADC. The flag ADRDY is set once the ADC is ready for operation.
- ADDIS = 1 disables the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART = 1 (refer to [Section 16.4.18: Conversion on external trigger and trigger polarity \(EXTSEL, EXTE, JEXTSEL, JEXTEN\)](#)) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART = 1 or when an external injected trigger event occurs, if injected triggers are enabled.

##### Software procedure to enable the ADC

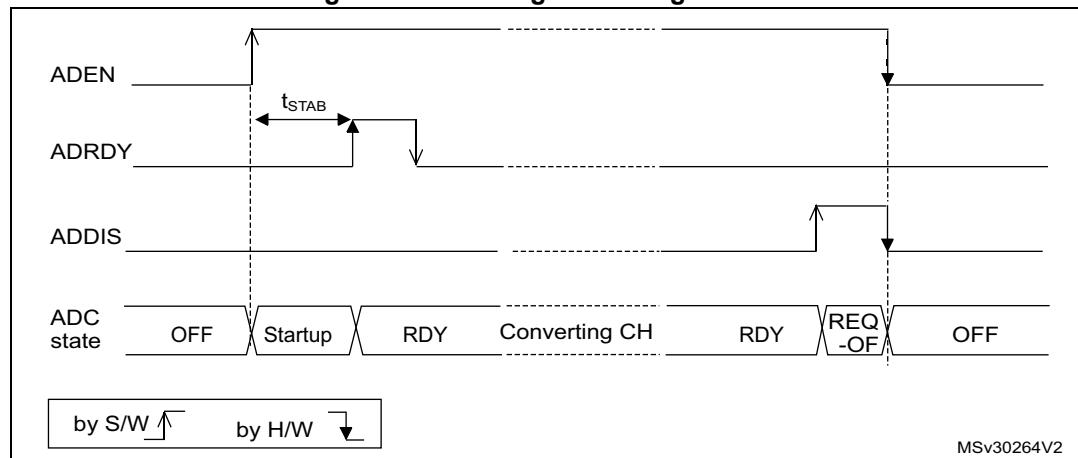
1. Clear the ADRDY bit in the ADC\_ISR register by writing 1.
2. Set ADEN.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE = 1).
4. Clear the ADRDY bit in the ADC\_ISR register by writing 1 (optional).

**Caution:** ADEN bit cannot be set when ADCAL is set and during four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration).

### Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP = 1 and JADSTP = 1 and then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS.
3. If required by the application, wait until ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 46. Enabling / disabling the ADC



#### 16.4.10 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the DIFSEL[i] control bits in the ADC\_DIFSEL register and the control bits ADCAL and ADEN in the ADC\_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC\_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC\_CFGR, ADC\_SMPRx, ADC\_SQRy, ADC\_JDRy, ADC\_OFRy, ADC\_OFCHRy and ADC\_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).

The software is allowed to write the ADSTP or JADSTP control bits of the ADC\_CR register only if the ADC is enabled, possibly converting, and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADC\_JSQR at any time, when the ADC is enabled (ADEN = 1). Refer to [Section 16.7.16: ADC injected sequence register \(ADC\\_JSQR\)](#) for additional details.

**Note:** *There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN as well as all the bits of ADC\_CR register).*

### 16.4.11 Channel selection (SQRx, JSQRx)

There are up to 19 multiplexed channels:

- Up to 11 slow analog inputs coming from GPIO pads (ADC1\_INP/INN[6:16])  
Depending on the products, not all of them are available on GPIO pads.
- The ADC is connected to the following internal analog inputs:
  - The internal reference voltage ( $V_{REFINT}$ ) is connected to ADC1\_INP0.
  - The internal temperature sensor ( $V_{TS}$ ) is connected to ADC1\_INP17.
  - The  $V_{BAT}$  monitoring channel ( $V_{BAT}/3$ ) is connected to ADC1\_INP18.

**Note:** *To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, CH17SEL or CH18SEL in the ADCx\_CCR registers.*

It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADC1\_INP/INN3, ADC1\_INP/INN8, ADC1\_INP/INN2, ADC1\_INN/INP2, ADC1\_INP/INN0, ADC1\_INP/INN2, ADC1\_INP/INN2, ADC1\_INP/INN15.

- 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\_SQRy 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.

ADC\_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP = 1 (refer to [Section 16.4.17: Stopping an ongoing conversion \(ADSTP, JADSTP\)](#)).

The software is allowed to modify on-the-fly the ADC\_JSQR register when JADSTART is set (injected conversions ongoing) only when the context queue is enabled (JQDIS = 0 in ADC\_CFGR register). Refer to [Section 16.4.21: Queue of context for injected conversions](#)

### 16.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC\_SMPR1 and ADC registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 6.5 ADC clock cycles
- SMP = 010: 12.5 ADC clock cycles
- SMP = 011: 24.5 ADC clock cycles
- SMP = 100: 47.5 ADC clock cycles
- SMP = 101: 92.5 ADC clock cycles
- SMP = 110: 247.5 ADC clock cycles
- SMP = 111: 640.5 ADC clock cycles

The total conversion time is calculated as follows:

$$T_{\text{CONV}} = \text{Sampling time} + 12.5 \text{ ADC clock cycles}$$

Example:

With  $F_{\text{ADC\_CLK}} = 30 \text{ MHz}$  and a sampling time of 2.5 ADC clock cycles:

$$T_{\text{CONV}} = (2.5 + 12.5) \text{ ADC clock cycles} = 15 \text{ ADC clock cycles} = 500 \text{ ns}$$

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

#### Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.

#### I/O analog switches voltage booster

The I/O analog switches resistance increases when the  $V_{\text{DDA}}$  voltage is too low. This requires to have the sampling time adapted accordingly (cf datasheet for electrical characteristics). This resistance can be minimized at low  $V_{\text{DDA}}$  by enabling an internal voltage booster with BOOSTEN bit in the SYSCFG\_CFGR1 register.

### 16.4.13 Single conversion mode (CONT = 0)

In Single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC\_CR register (for a regular channel)
- Setting the JADSTART bit in the ADC\_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC\_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 16-bit ADC\_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

*Note:* *To convert a single channel, program a sequence with a length of 1.*

#### 16.4.14 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC\_CR register.

Inside the regular sequence, after each conversion is complete:

- 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

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

*Note:* *To convert a single channel, program a sequence with a length of 1.*

*It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.*

*Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to [Auto-injection mode](#) section).*

### 16.4.15 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART = 1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTN[1:0] = 00 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTN[1:0] is not equal to 00

Software starts ADC injected conversions by setting JADSTART = 1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN[1:0] = 00 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN[1:0] is not equal to 00

**Note:** *In auto-injection mode (JAUTO = 1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).*

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART = 0 and JADSTART = 0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software regular trigger (CONT = 0, EXTSEL = 0x0)
  - At any end of regular conversion sequence (EOS assertion) or at any end of subgroup processing if DISCEN = 1
- In all cases (CONT=x, EXTSEL=x)
  - After execution of the ADSTP procedure asserted by the software.

**Note:** *In continuous mode (CONT = 1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.*

*When a hardware trigger is selected in single mode (CONT = 0 and EXTSEL≠0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.*

JADSTART is cleared by hardware:

- In single mode with software injected trigger (JEXTSEL = 0x0)
  - At any end of injected conversion sequence (JEOS assertion) or at any end of subgroup processing if JDISCEN = 1
- in all cases (JEXTSEL=x)
  - After execution of the JADSTP procedure asserted by the software.

**Note:** *When the software trigger is selected, ADSTART bit should not be set if the EOC flag is still high.*

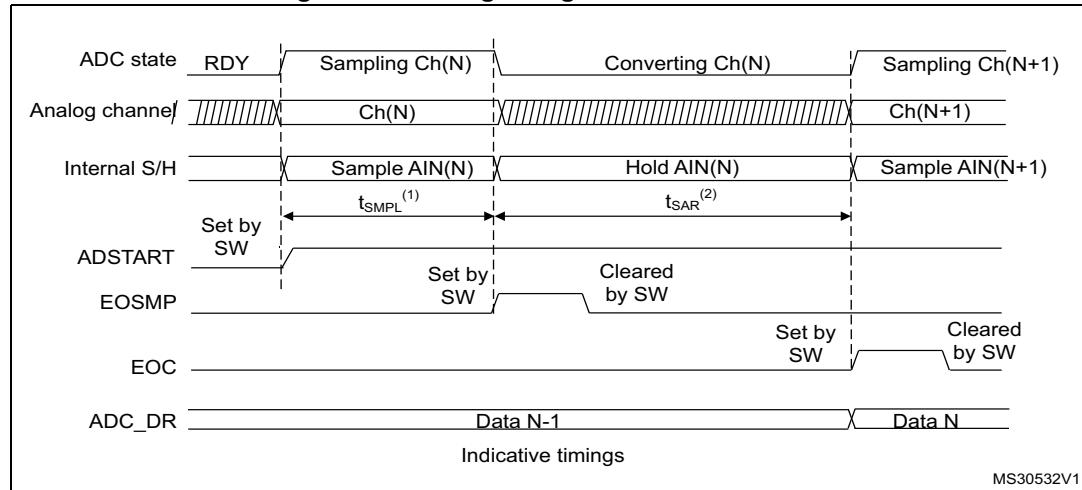
### 16.4.16 ADC timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$$T_{\text{CONV}} = T_{\text{SMPL}} + T_{\text{SAR}} = [2.5 \text{ } \mu\text{min} + 12.5 \text{ } \mu\text{min}] \times T_{\text{ADC\_CLK}}$$

$$T_{\text{CONV}} = T_{\text{SMPL}} + T_{\text{SAR}} = 83.33 \text{ ns } \mu\text{min} + 416.67 \text{ ns } \mu\text{min} = 500.0 \text{ ns} \text{ (for } F_{\text{ADC\_CLK}} = 30 \text{ MHz)}$$

Figure 47. Analog to digital conversion time



1.  $T_{\text{SMPL}}$  depends on SMP[2:0].

2.  $T_{\text{SAR}}$  depends on RES[2:0].

### 16.4.17 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP = 1 and injected conversions ongoing by setting JADSTP = 1.

Stopping conversions resets the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC\_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC\_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would restart a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In auto-injection mode ( $JAUTO = 1$ ), setting  $ADSTP$  bit aborts both regular and injected conversions ( $JADSTP$  must not be used).

Figure 48. Stopping ongoing regular conversions

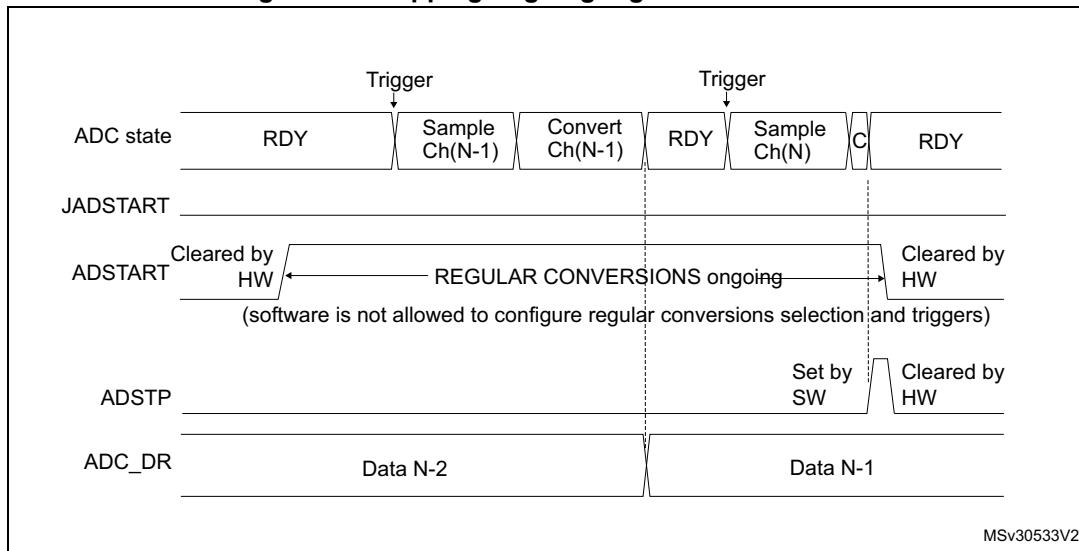
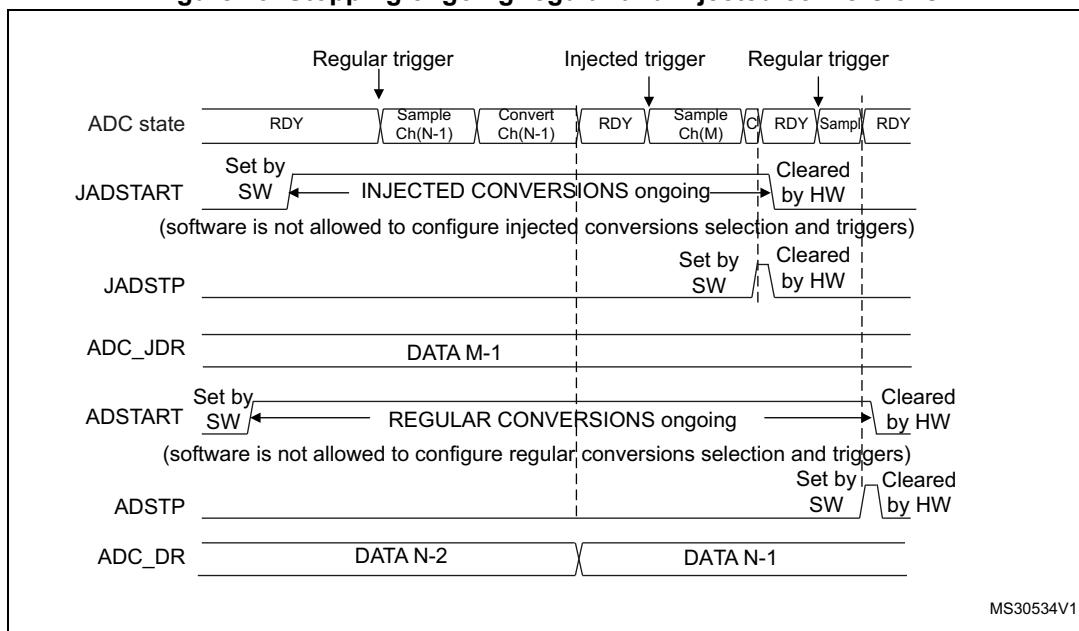


Figure 49. Stopping ongoing regular and injected conversions



### 16.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (e.g. timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS = 0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART = 1 and the injected trigger selection is effective once software has set bit JADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART = 0, any regular hardware triggers which occur are ignored.
- If bit JADSTART = 0, any injected hardware triggers which occur are ignored.

*Table 77* provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

**Table 77. Configuring the trigger polarity for regular external triggers**

EXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

*Note:* The polarity of the regular trigger cannot be changed on-the-fly.

**Table 78. Configuring the trigger polarity for injected external triggers**

JEXTEN[1:0]	Source
00	<ul style="list-style-type: none"> <li>– If JQDIS = 1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled</li> <li>– If JQDIS = 0 (Queue enabled), Hardware and software trigger detection disabled</li> </ul>
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

*Note:* The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS = 0). Refer to [Section 16.4.21: Queue of context for injected conversions](#).

The EXTSEL and JEXTSEL control bits select which out of 16 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

**Note:** *The regular trigger selection cannot be changed on-the-fly.  
The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 16.4.21: Queue of context for injected conversions on page 439*

*Table 79* to *Table 80* give all the possible external triggers of the three ADCs for regular and injected conversions.

**Table 79. ADC1 - External triggers for regular channels**

Name	Source	Type	EXTSEL[3:0]
EXT0	TIM1_CC1	Internal signal from on-chip timers	0000
EXT1	TIM1_CC2	Internal signal from on-chip timers	0001
EXT2	TIM1_CC3	Internal signal from on-chip timers	0010
EXT3	TIM2_CC2	Internal signal from on-chip timers	0011
EXT4	-	-	0100
EXT5	-	-	0101
EXT6	EXTI Line 11	External pin	0110
EXT9	TIM1_TRGO	Internal signal from on-chip timers	1001
EXT10	TIM1_TRGO2	Internal signal from on-chip timers	1010
EXT11	TIM2_TRGO	Internal signal from on-chip timers	1011
EXT13	-	-	1101
EXT14	-	-	1110

**Table 80. ADC1 - External trigger for injected channels**

Name	Source	Type	JEXTSEL[3:0]
JEXT0	TIM1_TRGO	Internal signal from on-chip timers	0000
JEXT1	TIM1_CC4	Internal signal from on-chip timers	0001
JEXT2	TIM2_TRGO	Internal signal from on-chip timers	0010
JEXT3	TIM2_CH1	Internal signal from on-chip timers	0011
JEXT4	-	-	0100
JEXT5	-	-	0101
JEXT6	EXTI Line 15	External pin	0110
JEXT8	TIM1_TRGO2	Internal signal from on-chip timers	1000
JEXT14	-	-	1110
JEXT15	-	-	1111

### 16.4.19 Injected channel management

#### Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC\_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC\_CR register.
  2. If an external injected trigger occurs, or if the JADSTART bit in the ADC\_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once).
  3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
  4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence.
- Figure 50* 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 2.5 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.*

#### Auto-injection mode

If the JAUTO bit in the ADC\_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC\_SQRy and ADC\_JSQR registers.

In this mode, the ADSTART bit in the ADC\_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

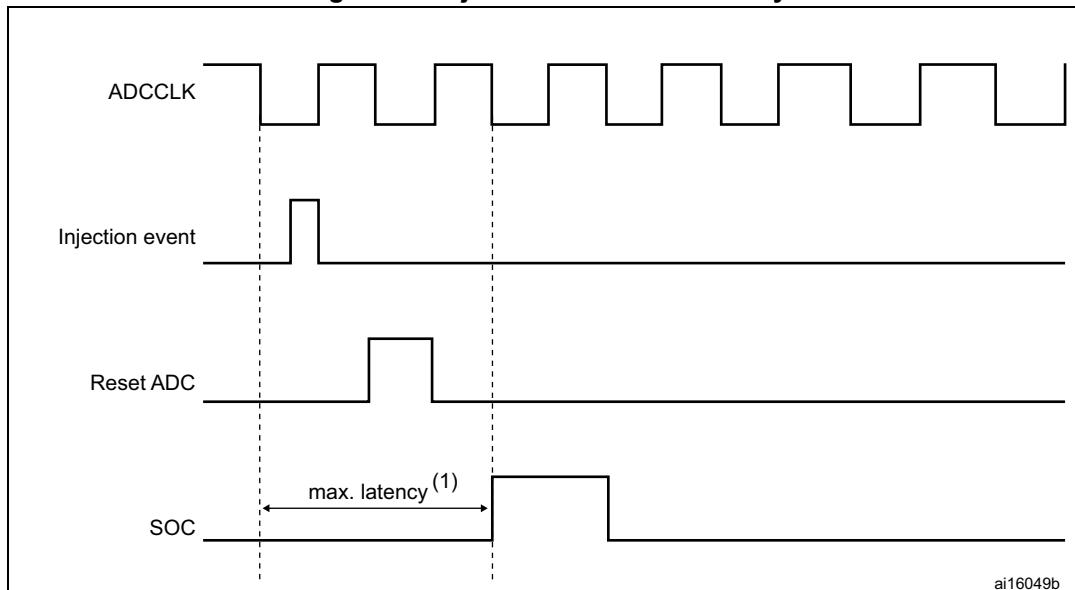
In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

**Note:** *It is not possible to use both the auto-injected and discontinuous modes simultaneously.*

*When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA\_CCRx register). If the CIRC bit is reset (single-shot mode), the JAUTO sequence is stopped upon DMA Transfer Complete event.*

Figure 50. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

#### 16.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

##### Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC\_CFGR register.

It is used to convert a short sequence (subgroup) of  $n$  conversions ( $n \leq 8$ ) that is part of the sequence of conversions selected in the ADC\_SQRy registers. The value of  $n$  is specified by writing to the DISCNUM[2:0] bits in the ADC\_CFGR register.

When an external trigger occurs, it starts the next  $n$  conversions selected in the ADC\_SQRy 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:

- DISCEN = 1,  $n = 3$ , channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

**Note:** *The channel numbers referred to in the above example might not be available on all microcontrollers.*

*When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).*

*When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.*

*It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.*

### Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC\_CFGR register. It converts the sequence selected in the ADC\_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where 'n' is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the 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:**

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

**Note:** *The channel numbers referred to in the above example might not be available on all microcontrollers.*

*When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.*

*It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.*

### 16.4.21 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC\_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in ADC\_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC\_JSQR register)

All the parameters of the context are defined into a single register ADC\_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC\_CFGR:
  - If JQM = 0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence is served according to the last active context.
  - If JQM = 1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP = 1 or when disabling the ADC by setting ADDIS = 1:
  - If JQM = 0, the Queue is maintained with the last active context.
  - If JQM = 1, the Queue becomes empty and triggers are ignored.

Note:

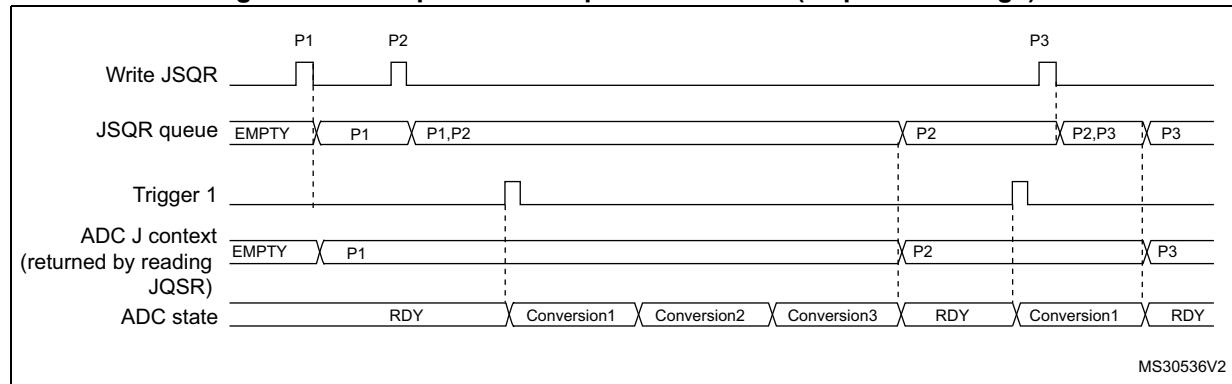
*When configured in discontinuous mode (bit JDISCEN = 1), only the last trigger of the injected sequence changes the context and consumes the Queue. The 1<sup>st</sup> trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):*

- 1<sup>st</sup> trigger, discontinuous. Sequence 1: context 1 consumed, 1<sup>st</sup> conversion carried out
- 2<sup>nd</sup> trigger, discontinuous. Sequence 1: 2<sup>nd</sup> conversion.
- 3<sup>rd</sup> trigger, discontinuous. Sequence 1: 3<sup>rd</sup> conversion.
- 4<sup>th</sup> trigger, discontinuous. Sequence 2: context 2 consumed, 1<sup>st</sup> conversion carried out.
- 5<sup>th</sup> trigger, discontinuous. Sequence 2: 2<sup>nd</sup> conversion.
- 6<sup>th</sup> trigger, discontinuous. Sequence 2: 3<sup>rd</sup> conversion.

### Behavior when changing the trigger or sequence context

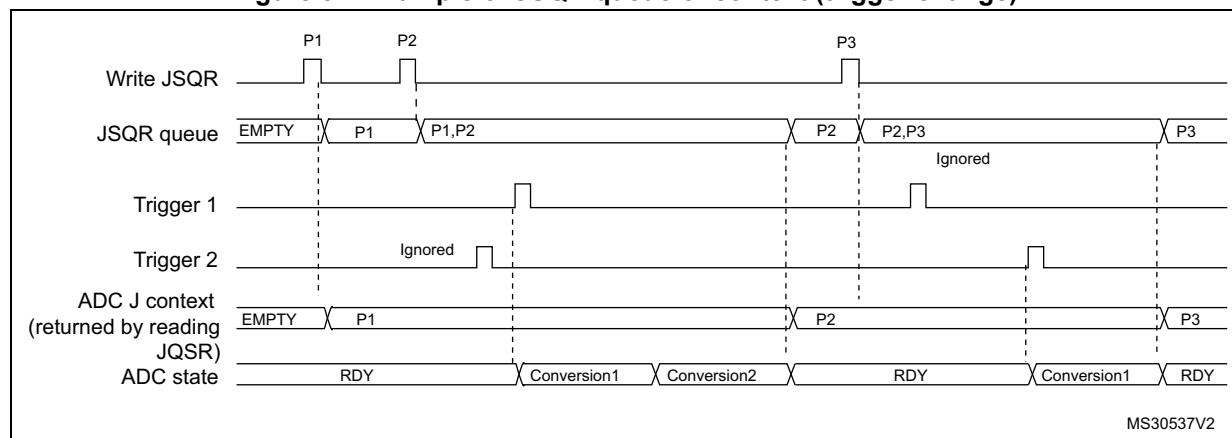
The [Figure 51](#) and [Figure 52](#) show the behavior of the context Queue when changing the sequence or the triggers.

**Figure 51. Example of JSQR queue of context (sequence change)**



1. Parameters:  
 P1: sequence of 3 conversions, hardware trigger 1  
 P2: sequence of 1 conversion, hardware trigger 1  
 P3: sequence of 4 conversions, hardware trigger 1

**Figure 52. Example of JSQR queue of context (trigger change)**

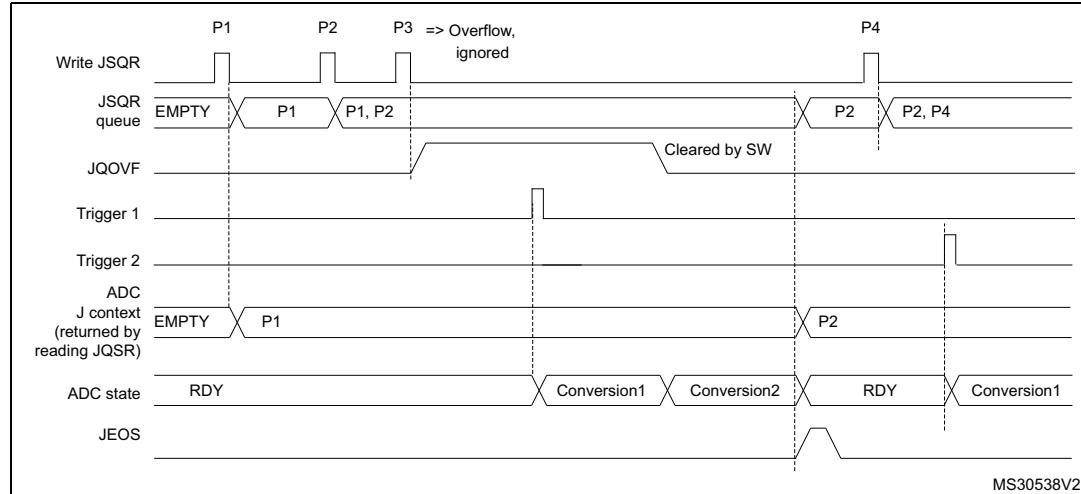


1. Parameters:  
 P1: sequence of 2 conversions, hardware trigger 1  
 P2: sequence of 1 conversion, hardware trigger 2  
 P3: sequence of 4 conversions, hardware trigger 1

### Queue of context: Behavior when a queue overflow occurs

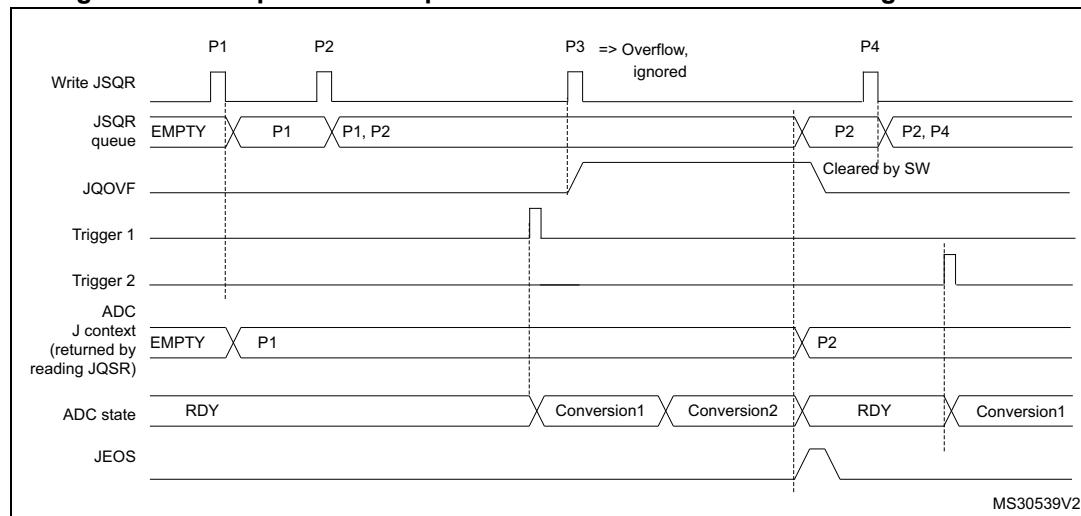
The [Figure 53](#) and [Figure 54](#) show the behavior of the context Queue if an overflow occurs before or during a conversion.

**Figure 53. Example of JSQR queue of context with overflow before conversion**



1. Parameters:
  - P1: sequence of 2 conversions, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 2
  - P3: sequence of 3 conversions, hardware trigger 1
  - P4: sequence of 4 conversions, hardware trigger 1

**Figure 54. Example of JSQR queue of context with overflow during conversion**



1. Parameters:
  - P1: sequence of 2 conversions, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 2
  - P3: sequence of 3 conversions, hardware trigger 1
  - P4: sequence of 4 conversions, hardware trigger 1

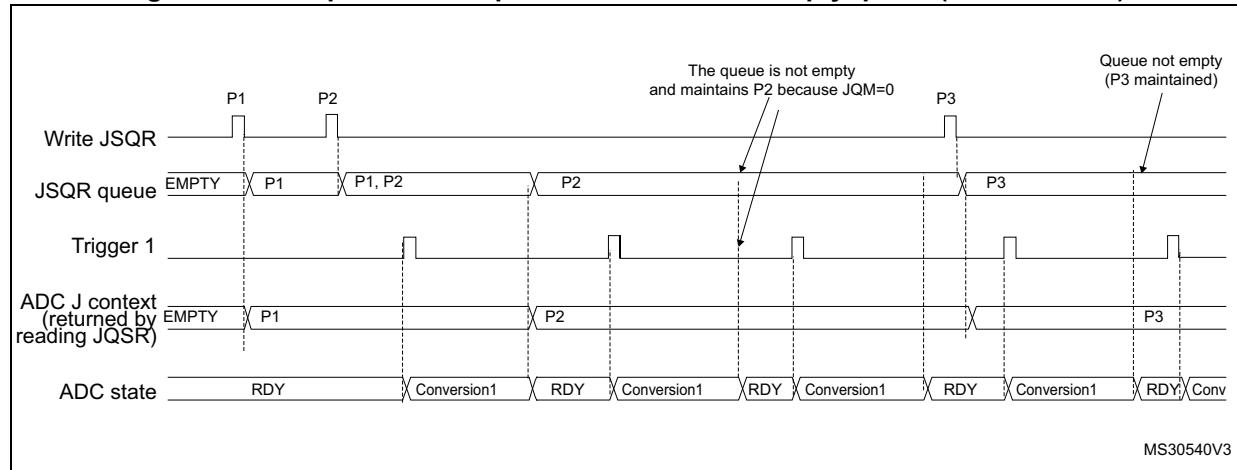
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

### Queue of context: Behavior when the queue becomes empty

*Figure 55* and *Figure 56* show the behavior of the context Queue when the Queue becomes empty in both cases  $JQM = 0$  or  $1$ .

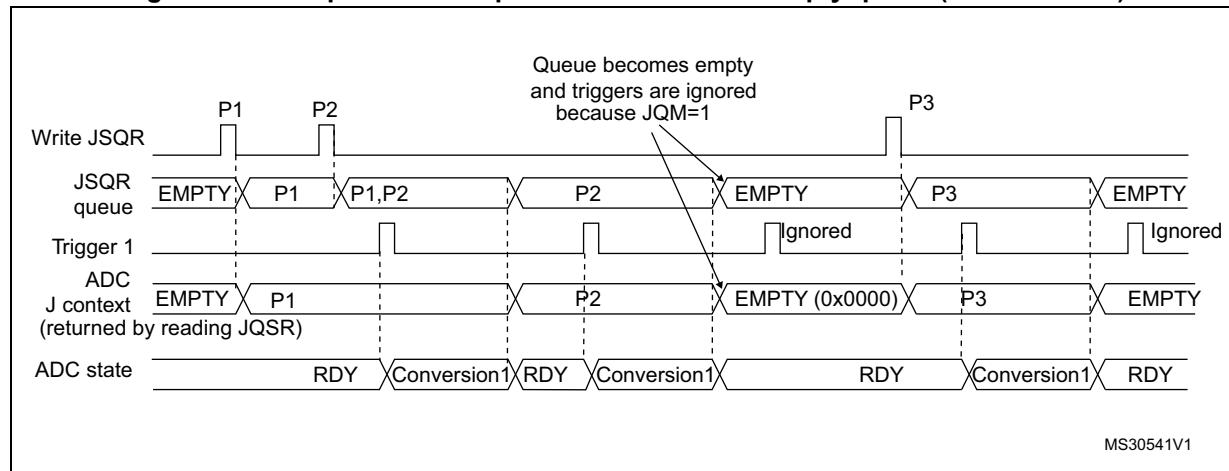
**Figure 55. Example of JSQR queue of context with empty queue (case  $JQM = 0$ )**



1. Parameters:
  - P1: sequence of 1 conversion, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 1
  - P3: sequence of 1 conversion, hardware trigger 1

**Note:** *When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.*

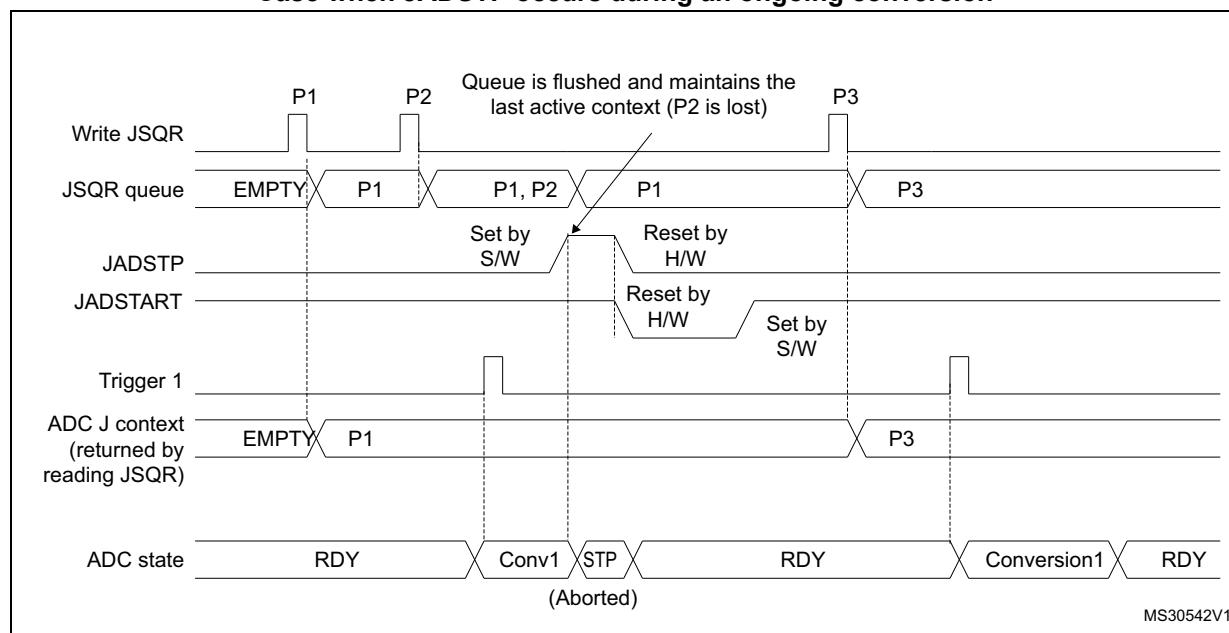
Figure 56. Example of JSQR queue of context with empty queue (case JQM = 1)



- Parameters:
  - P1: sequence of 1 conversion, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 1
  - P3: sequence of 1 conversion, hardware trigger 1

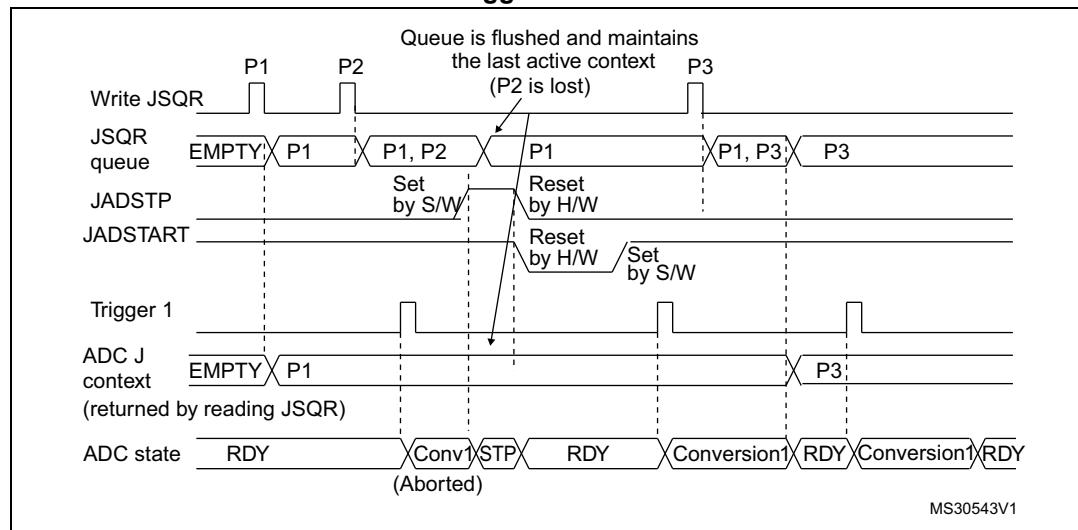
### Flushing the queue of context

The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 57. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0)  
Case when JADSTP occurs during an ongoing conversion

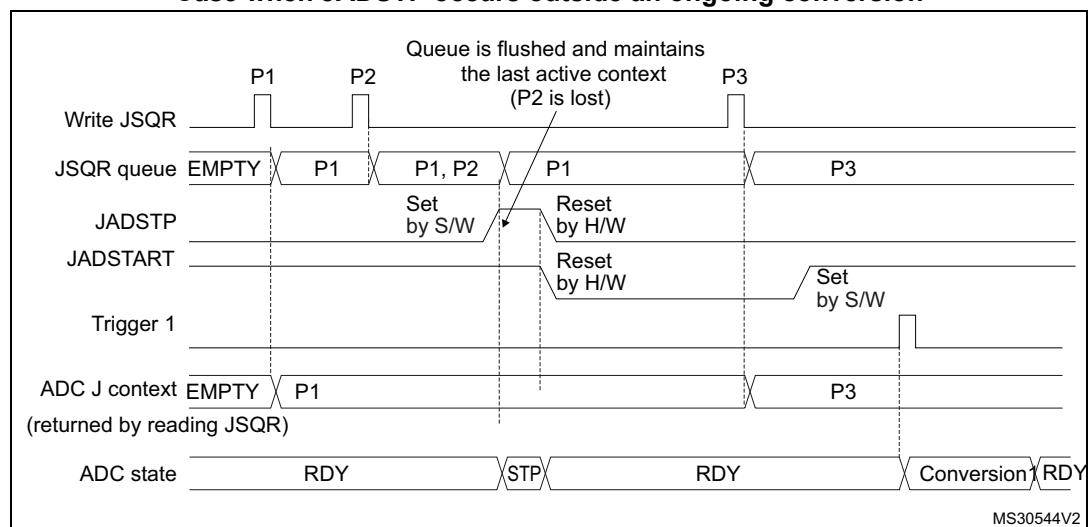
- Parameters:
  - P1: sequence of 1 conversion, hardware trigger 1
  - P2: sequence of 1 conversion, hardware trigger 1
  - P3: sequence of 1 conversion, hardware trigger 1

**Figure 58. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0)**  
**Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.**



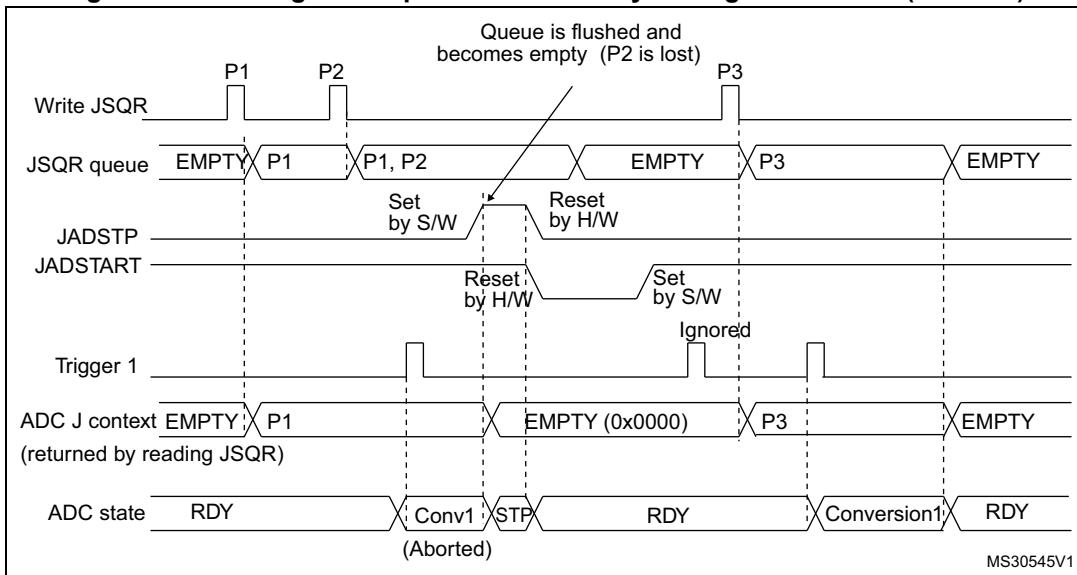
1. Parameters:  
P1: sequence of 1 conversion, hardware trigger 1  
P2: sequence of 1 conversion, hardware trigger 1  
P3: sequence of 1 conversion, hardware trigger 1

**Figure 59. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0)**  
**Case when JADSTP occurs outside an ongoing conversion**



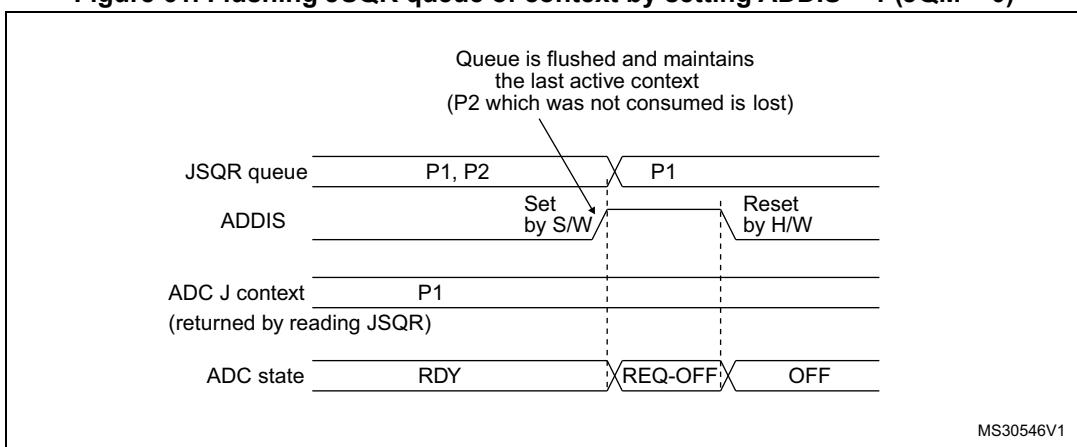
1. Parameters:  
P1: sequence of 1 conversion, hardware trigger 1  
P2: sequence of 1 conversion, hardware trigger 1  
P3: sequence of 1 conversion, hardware trigger 1

Figure 60. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1)

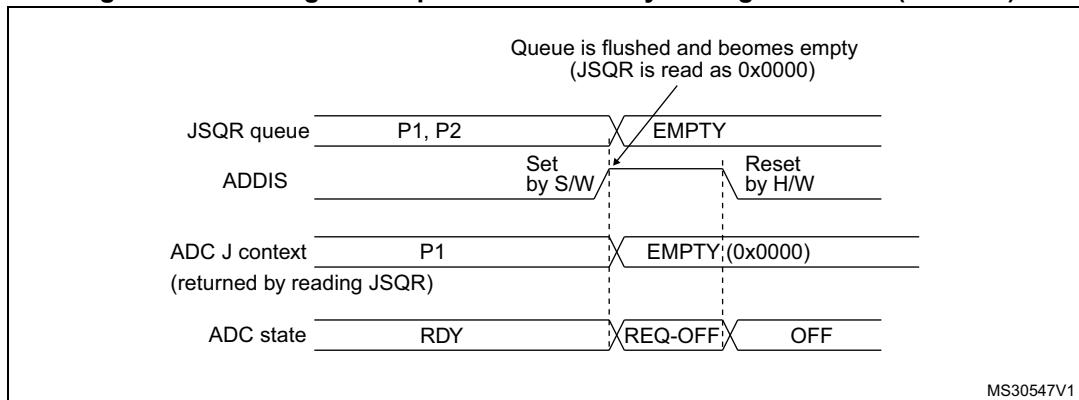


1. Parameters:  
 P1: sequence of 1 conversion, hardware trigger 1  
 P2: sequence of 1 conversion, hardware trigger 1  
 P3: sequence of 1 conversion, hardware trigger 1

Figure 61. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0)



1. Parameters:  
 P1: sequence of 1 conversion, hardware trigger 1  
 P2: sequence of 1 conversion, hardware trigger 1  
 P3: sequence of 1 conversion, hardware trigger 1

**Figure 62. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 1)**

1. Parameters:  
P1: sequence of 1 conversion, hardware trigger 1  
P2: sequence of 1 conversion, hardware trigger 1  
P3: sequence of 1 conversion, hardware trigger 1

### Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy JSQR with JEXTEN[1:0] not equal to 00 (otherwise triggering a software conversion).
6. Set JADSTART.
7. Set JADSTP.
8. Wait until JADSTART is reset.
9. Set JADSTART.

### Disabling the queue

It is possible to disable the queue by setting bit JQDIS = 1 into the ADC\_CFGR register.

#### 16.4.22 Programmable resolution (RES) - Fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the control bits RES[1:0]. [Figure 67](#), [Figure 68](#), [Figure 69](#) and [Figure 70](#) show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to [Table 81](#).

**Table 81.  $T_{SAR}$  timings depending on resolution**

RES (bits)	$T_{SAR}$ (ADC clock cycles)	$T_{SAR}$ (ns) at $F_{ADC} = 30$ MHz	$T_{CONV}$ (ADC clock cycles) (with Sampling Time = 2.5 ADC clock cycles)	$T_{CONV}$ (ns) at $F_{ADC} = 30$ MHz
12	12.5 ADC clock cycles	416.67 ns	15 ADC clock cycles	500.0 ns
10	10.5 ADC clock cycles	350.0 ns	13 ADC clock cycles	433.33 ns
8	8.5 ADC clock cycles	203.33 ns	11 ADC clock cycles	366.67 ns
6	6.5 ADC clock cycles	216.67 ns	9 ADC clock cycles	300.0 ns

#### 16.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC\_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC\_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC\_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC\_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

#### 16.4.24 End of conversion sequence (EOS, JEOS)

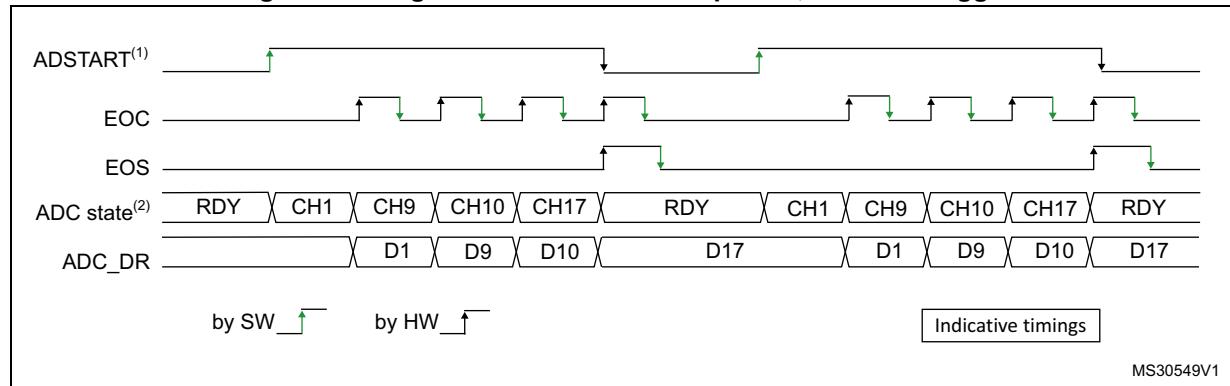
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC\_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

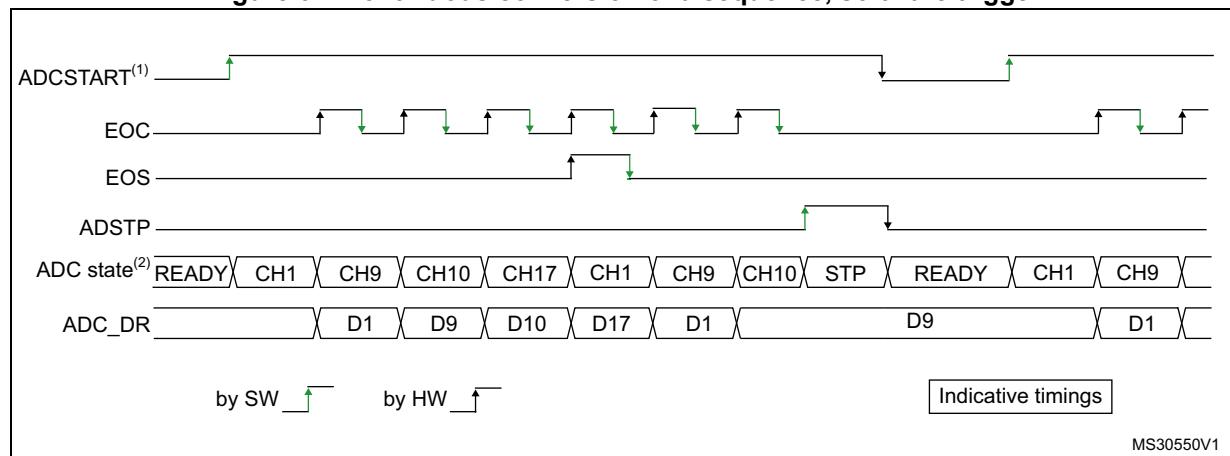
### 16.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

Figure 63. Single conversions of a sequence, software trigger



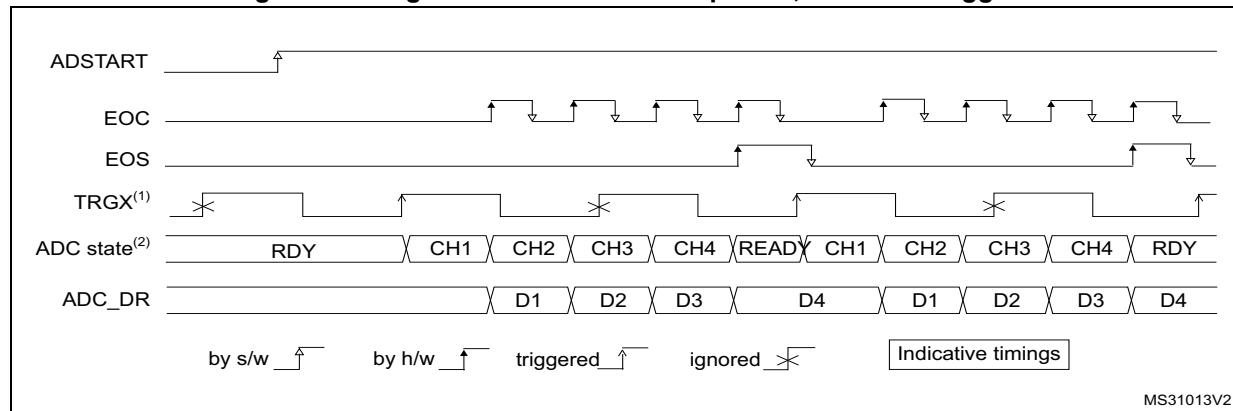
1. EXLEN[1:0] = 00, CONT = 0
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

Figure 64. Continuous conversion of a sequence, software trigger



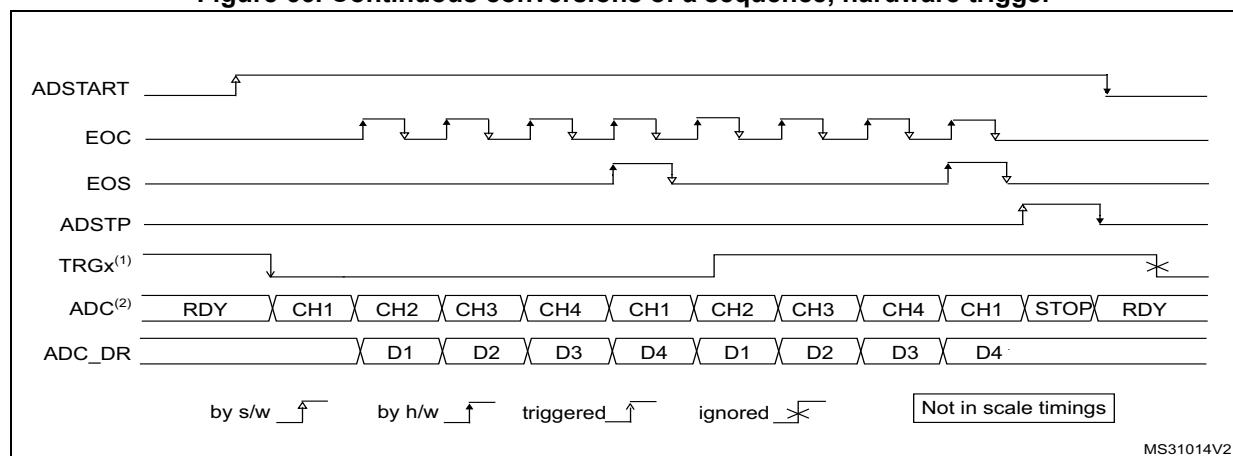
1. EXLEN[1:0] = 00, CONT = 1
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

Figure 65. Single conversions of a sequence, hardware trigger



1. TRGx (over-frequency) is selected as trigger source, EXTEN[1:0] = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

Figure 66. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN[1:0] = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

### 16.4.26 Data management

#### Data register, data alignment and offset (ADC\_DR, OFFSETy, OFFSETy\_CH, ALIGN)

##### Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADC\_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADC\_JDRy data register which is 16 bits wide.

The ALIGN bit in the ADC\_CFGR register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in [Figure 67](#), [Figure 68](#), [Figure 69](#) and [Figure 70](#).

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 69](#) and [Figure 70](#).

*Note:* *Left-alignment is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the ALIGN bit value is ignored and the ADC only provides right-aligned data.*

##### Offset

An offset y (y = 1,2,3,4) can be applied to a channel by setting the bit OFFSETy\_EN = 1 into ADC\_OF Ry register. The channel to which the offset is applied is programmed into the bits OFFSETy\_CH[4:0] of ADC\_OF Ry register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSETy[11:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

*Note:* *Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSETy\_EN bit in ADC\_OF Ry register is ignored (considered as reset).*

[Table 84](#) describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

**Table 82. Offset computation versus data resolution**

Resolution (bits RES[1:0])	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	Signed 12-bit data	-
01: 10-bit	DATA[11:2],00	OFFSET[11:0]	Signed 10-bit data	The user must configure OFFSET[1:0] to 00

Table 82. Offset computation versus data resolution (continued)

Resolution (bits RES[1:0])	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
10: 8-bit	DATA[11:4],00 00	OFFSET[11:0]	Signed 8-bit data	The user must configure OFFSET[3:0] to 0000
11: 6-bit	DATA[11:6],00 0000	OFFSET[11:0]	Signed 6-bit data	The user must configure OFFSET[5:0] to 000000

When reading data from ADC\_DR (regular channel) or from ADC\_JDRy (injected channel,  $y = 1,2,3,4$ ) corresponding to the channel "i":

- If one of the offsets is enabled (bit OFFSETy\_EN = 1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

*Figure 67, Figure 68, Figure 69* and *Figure 70* show alignments for signed and unsigned data.

Figure 67. Right alignment (offset disabled, unsigned value)

<u>12-bit data</u>	bit15	bit7	bit0
0 0 0 0 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0			
<u>10-bit data</u>	bit15	bit7	bit0
0 0 0 0 0 0 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0			
<u>8-bit data</u>	bit15	bit7	bit0
0 0 0 0 0 0 0 D7 D6 D5 D4 D3 D2 D1 D0			
<u>6-bit data</u>	bit15	bit7	bit0
0 0 0 0 0 0 0 0 0 D5 D4 D3 D2 D1 D0			

MS31015V1

**Figure 68. Right alignment (offset enabled, signed value)**

<u>12-bit data</u>																
bit15	bit7								bit0							
SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
<u>10-bit data</u>																
bit15	bit7								bit0							
SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
<u>8-bit data</u>																
bit15	bit7								bit0							
SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	D7	D6	D5	D4	D3	D2	D1	D0
<u>6-bit data</u>																
bit15	bit7								bit0							
SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	D5	D4	D3	D2	D1	D0		

**Figure 69. Left alignment (offset disabled, unsigned value)**

<u>12-bit data</u>															
bit15								bit7							
D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
<u>10-bit data</u>															
bit15								bit7							
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0
<u>8-bit data</u>															
bit15								bit7							
D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0	0
<u>6-bit data</u>															
bit15								bit7							
0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0	0	0

Figure 70. Left alignment (offset enabled, signed value)

<u>12-bit data</u>
bit15
SEXT D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 0 0 0
bit7
bit0
<u>10-bit data</u>
bit15
SEXT D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0 0
bit7
bit0
<u>8-bit data</u>
bit15
SEXT D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0 0 0 0
bit7
bit0
<u>6-bit data</u>
bit15
SEXT SEXT SEXT SEXT SEXT SEXT SEXT SEXT D5 D4 D3 D2 D1 D0 0
bit7
bit0

MS31018V1

### ADC overrun (OVR, OVRMOD)

The overrun flag (OSR) notifies of that a buffer overrun event occurred when the regular converted data has not been read (by the CPU or the DMA) before new converted data became available.

The OVR flag is set if the EOC flag is still 1 at the time when a new conversion completes. An interrupt can be generated if bit OVRIE = 1.

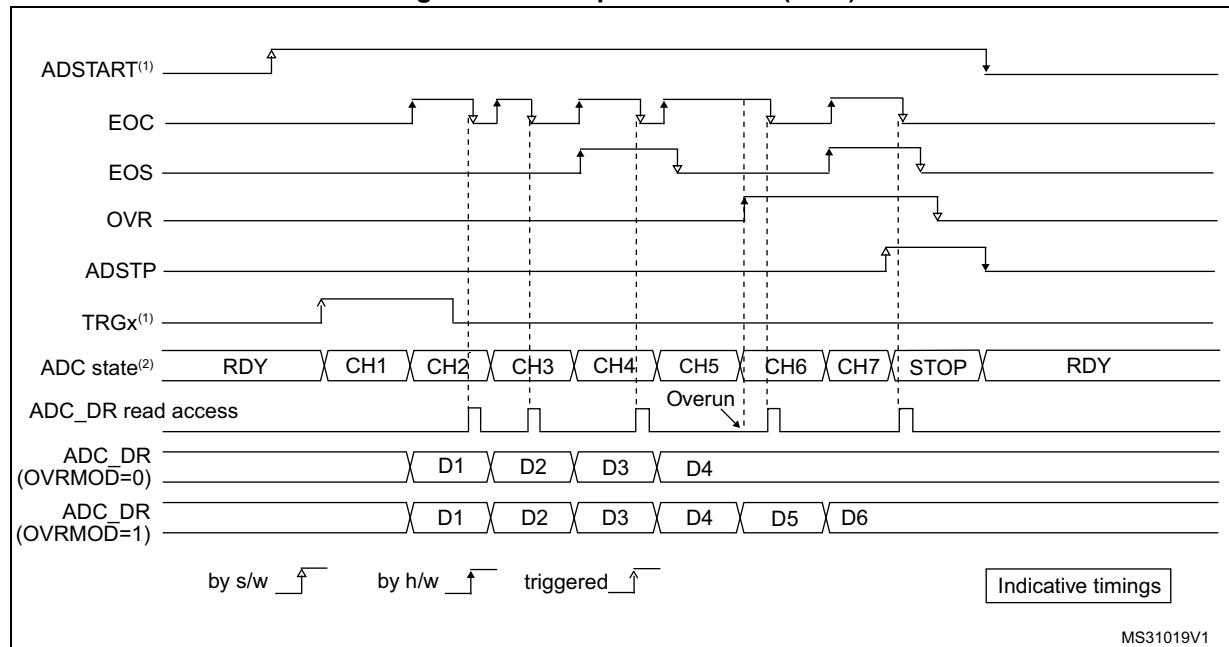
When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting bit ADSTP = 1.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0: The overrun event preserves the data register from being overrun: the old data is maintained and the new conversion is discarded and lost. If OVR remains at 1, any further conversions occur but the result data is also discarded.
- OVRMOD = 1: The data register is overwritten with the last conversion result and the previous unread data is lost. If OVR remains at 1, any further conversions operate normally and the ADC\_DR register always contains the latest converted data.

Figure 71. Example of overrun (OVR)



**Note:** There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

### Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC\_DR register can be read. OVRMOD should be configured to 0 to manage overrun events as an error.

### Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC\_DR register always contains the latest conversion.

### Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC\_DR register.

When the DMA mode is enabled (DMAEN bit set in the ADC\_CFGR register), a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC\_DR register to the destination location selected by the software.

Despite this, if an overrun occurs ( $OVR = 1$ ) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of  $OVRMOD$  bit, the data is either preserved or overwritten (refer to [Section : ADC overrun \( \$OVR\$ ,  \$OVRMOD\$ \)](#)).

The DMA transfer requests are blocked until the software clears the  $OVR$  bit.

Two different DMA modes are proposed depending on the application use and are configured with bit  $DMACFG$  of the  $ADC\_CFG$  register:

- DMA one shot mode ( $DMACFG = 0$ ).  
This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode ( $DMACFG = 1$ )  
This mode is suitable when programming the DMA in circular mode.

### DMA one shot mode ( $DMACFG = 0$ )

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs - refer to DMA section) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.

### DMA circular mode ( $DMACFG = 1$ )

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

## 16.4.27 Dynamic low-power features

### Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the  $AUTDLY$  configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When  $AUTDLY = 1$ , a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the  $ADC\_DR$  register has been read or if the  $EOC$  bit has been cleared (see [Figure 72](#)).
- For an injected conversion: when the  $JEOS$  bit has been cleared (see [Figure 73](#)).

This is a way to automatically adapt the speed of the ADC to the speed of the system which reads the data.

The delay is inserted after each regular conversion (whatever DISCEN = 0 or 1) and after each sequence of injected conversions (whatever JDISCEN = 0 or 1).

**Note:** *There is no delay inserted between each conversions of the injected sequence, except after the last one.*

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

**Note:** *This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to restart a conversion: it is up to the software to read the data before launching a new conversion.*

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see [Figure 73](#)).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see [Figure 75](#)).

The behavior is slightly different in auto-injected mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see [Figure 76](#)).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO = 1, CONT = 1 and AUTDLY = 1), follow the following procedure:

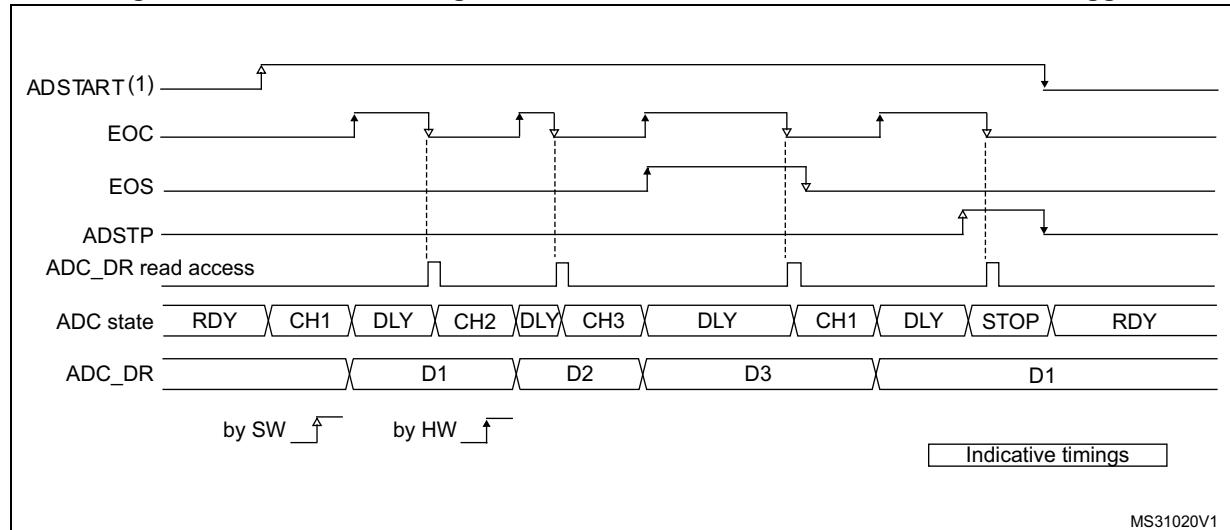
1. Wait until JEOS = 1 (no more conversions are restarted)
2. Clear JEOS.
3. Set ADSTP.
4. Read the regular data.

If this procedure is not respected, a new regular sequence can restart if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

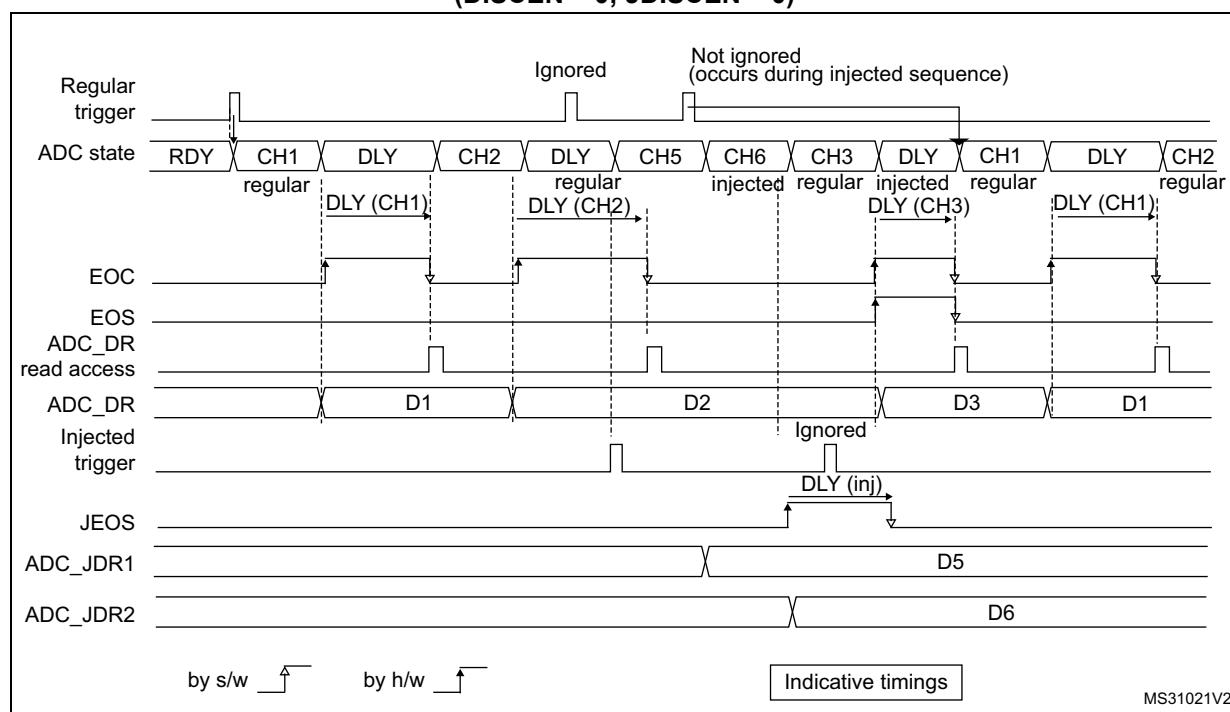
In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

Figure 72. AUTDLY = 1, regular conversion in continuous mode, software trigger



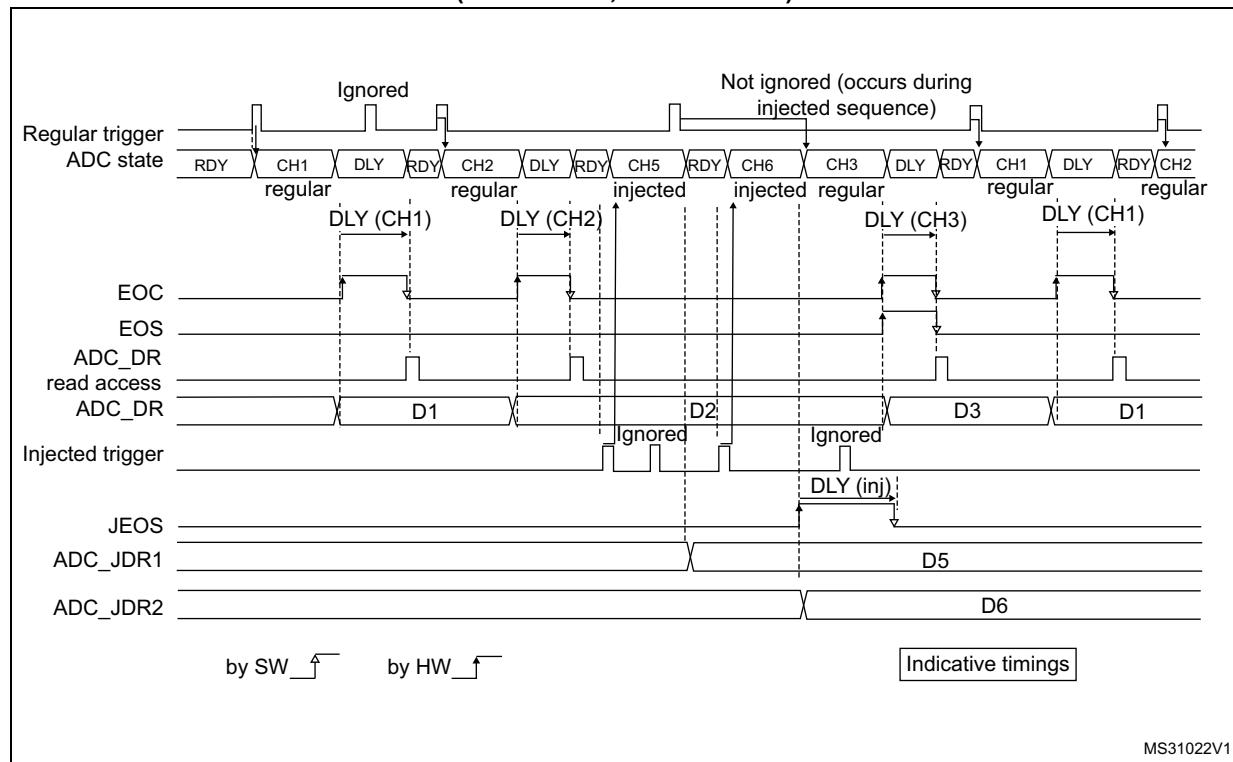
1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (SW trigger), CONT = 1, CHANNELS = 1,2,3.
3. Injected configuration DISABLED.

Figure 73. AUTDLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)



1. AUTDLY = 1
2. Regular configuration: EXTEN[1:0] = 01 (HW trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN[1:0] = 01 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6

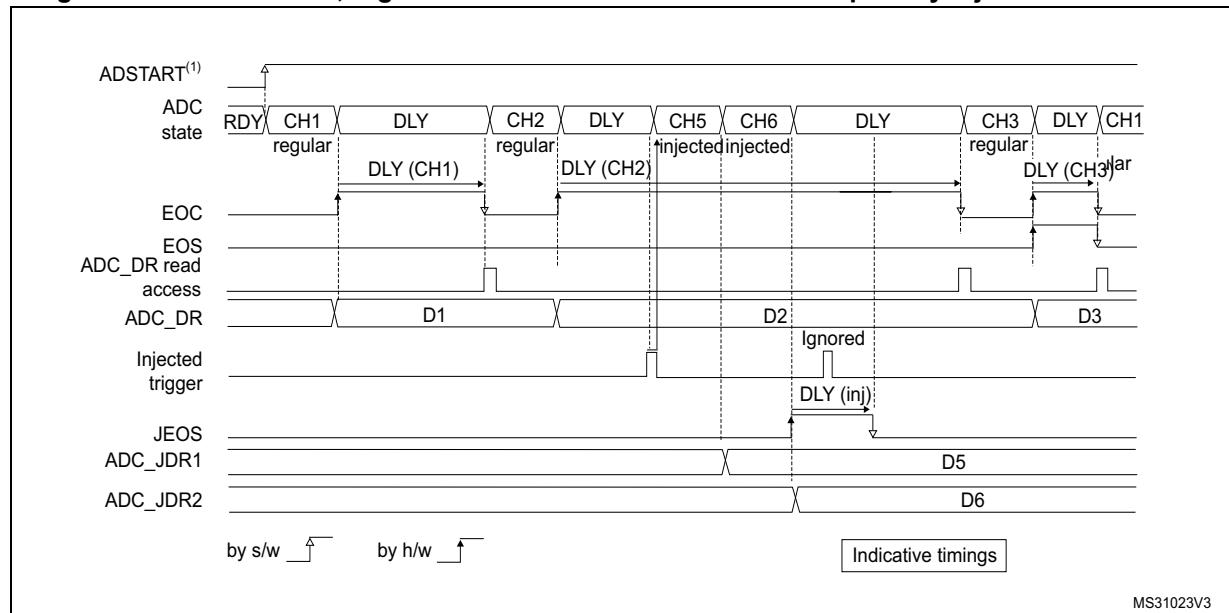
Figure 74. AUTODYL = 1, regular HW conversions interrupted by injected conversions  
(DISCEN = 1, JDISCEN = 1)



MS31022V1

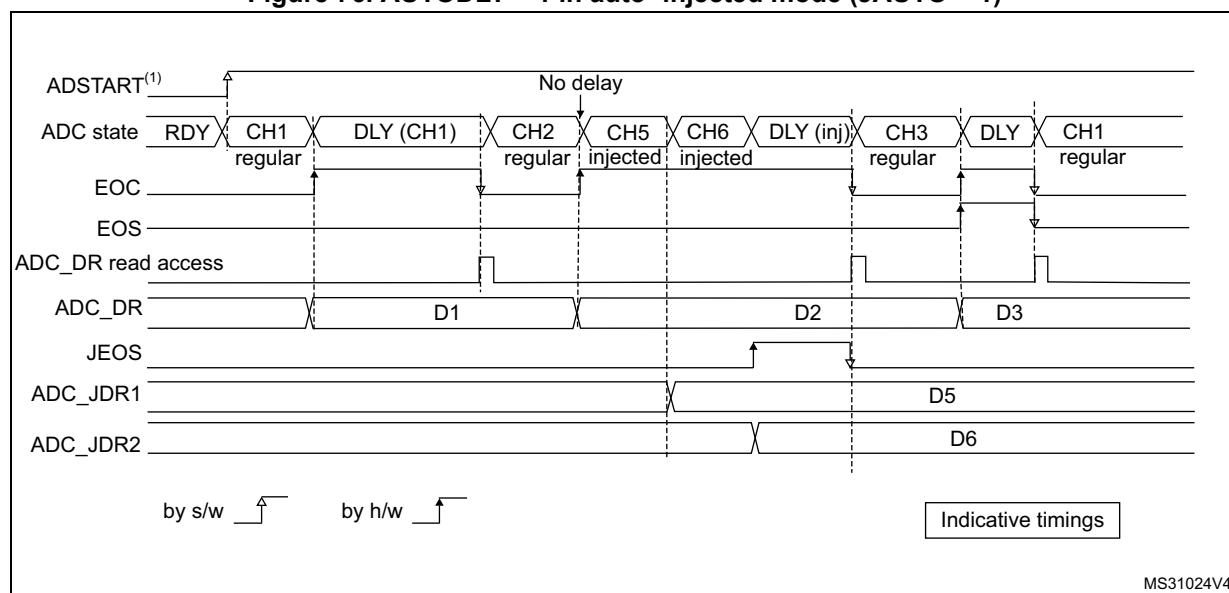
1. AUTODYL = 1
2. Regular configuration: EXTEN[1:0] = 01 (HW trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (HW Trigger), JDISCEN = 1, CHANNELS = 5,6

Figure 75. AUTODLY = 1, regular continuous conversions interrupted by injected conversions



1. AUTDLY = 1
2. Regular configuration: EXTEN[1:0] = 00 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN[1:0] = 01 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6

Figure 76. AUTODLY = 1 in auto- injected mode (JAUTO = 1)

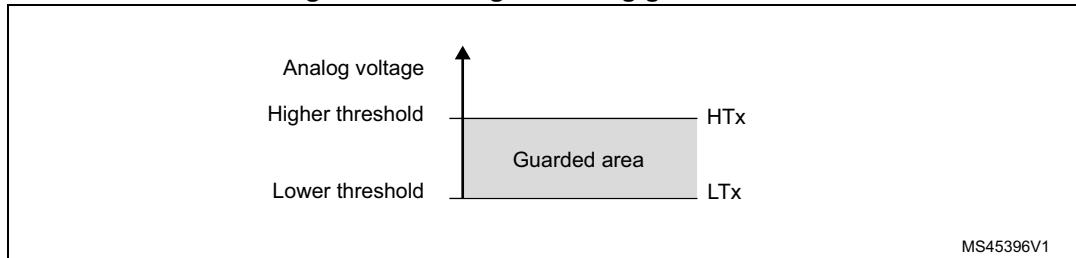


1. AUTDLY = 1
2. Regular configuration: EXTEN[1:0] = 00 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2
3. Injected configuration: JAUTO = 1, CHANNELS = 5,6

### 16.4.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD\_LTx, AWDx)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 77. Analog watchdog guarded area



#### AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDxIE in the ADC\_IER register (x = 1,2,3).

AWDx (x = 1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

#### Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC\_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels<sup>(1)</sup> remain within a configured voltage range (window).

*Table 83* shows how the ADC\_CFGR registers should be configured to enable the analog watchdog on one or more channels.

Table 83. Analog watchdog channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single <sup>(1)</sup> injected channel	1	0	1
Single <sup>(1)</sup> regular channel	1	1	0
Single <sup>(1)</sup> regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed in bits HT1[11:0] and LT1[11:0] of the ADC\_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits (according to bits RES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

[Table 84](#) describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

**Table 84. Analog watchdog 1 comparison**

Resolution bit RES[1:0]	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	Thresholds	
00: 12-bit	DATA[11:0]	LT1[11:0] and HT1[11:0]	-
01: 10-bit	DATA[11:2],00	LT1[11:0] and HT1[11:0]	User must configure LT1[1:0] and HT1[1:0] to 00
10: 8-bit	DATA[11:4],0000	LT1[11:0] and HT1[11:0]	User must configure LT1[3:0] and HT1[3:0] to 0000
11: 6-bit	DATA[11:6],000000	LT1[11:0] and HT1[11:0]	User must configure LT1[5:0] and HT1[5:0] to 000000

### Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDxCH[18:0] (x=2,3).

The corresponding watchdog is enabled when any bit of AWDxCH[18:0] (x=2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. [Table 85](#) describes how the comparison is performed for all the possible resolutions.

**Table 85. Analog watchdog 2 and 3 comparison**

Resolution (bits RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	Thresholds	
00: 12-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:0] are not relevant for the comparison
01: 10-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:2] are not relevant for the comparison
10: 8-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	-
11: 6-bit	DATA[11:6],00	LTx[7:0] and HTx[7:0]	User must configure LTx[1:0] and HTx[1:0] to 00

### ADC<sub>y</sub>\_AWD<sub>x</sub>\_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADC<sub>y</sub>\_AWD<sub>x</sub>\_OUT ( $y$ =ADC number,  $x$ =watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADC<sub>y</sub>\_AWD<sub>x</sub>\_OUT signal as ETR.

ADC<sub>y</sub>\_AWD<sub>x</sub>\_OUT is activated when the associated analog watchdog is enabled:

- ADC<sub>y</sub>\_AWD<sub>x</sub>\_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC<sub>y</sub>\_AWD<sub>x</sub>\_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADC<sub>y</sub>\_AWD<sub>x</sub>\_OUT is also reset when disabling the ADC (when setting ADDIS = 1). Note that stopping regular or injected conversions (setting ADSTP = 1 or JADSTP = 1) has no influence on the generation of ADC<sub>y</sub>\_AWD<sub>x</sub>\_OUT.

**Note:** *AWD<sub>x</sub> flag is set by hardware and reset by software: AWD<sub>x</sub> flag has no influence on the generation of ADC<sub>y</sub>\_AWD<sub>x</sub>\_OUT (ex: ADC<sub>y</sub>\_AWD<sub>x</sub>\_OUT can toggle while AWD<sub>x</sub> flag remains at 1 if the software did not clear the flag).*

**Figure 78. ADC<sub>y</sub>\_AWD<sub>x</sub>\_OUT signal generation (on all regular channels)**

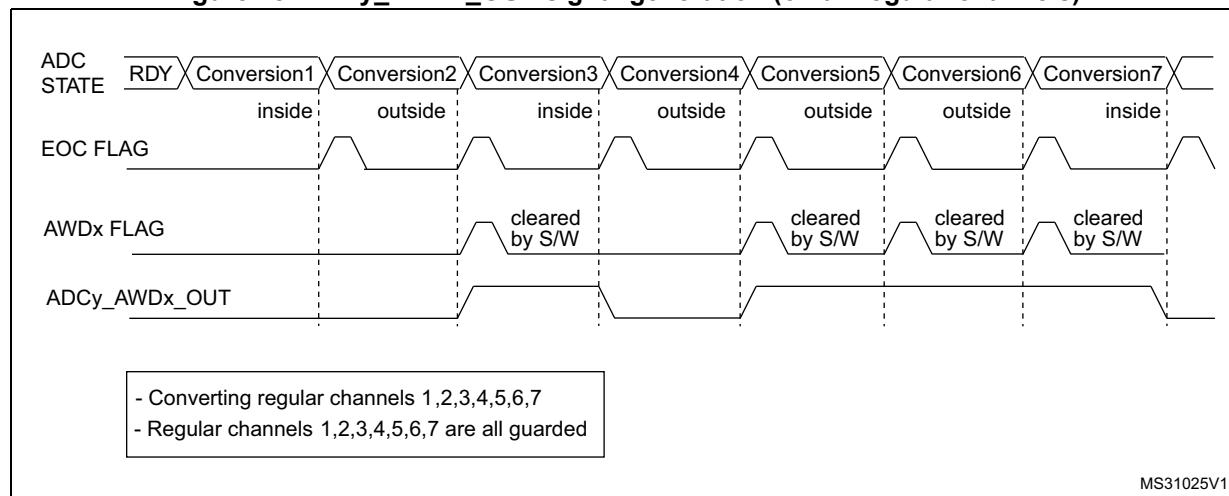


Figure 79. ADCy\_AWDx\_OUT signal generation (AWDx flag not cleared by software)

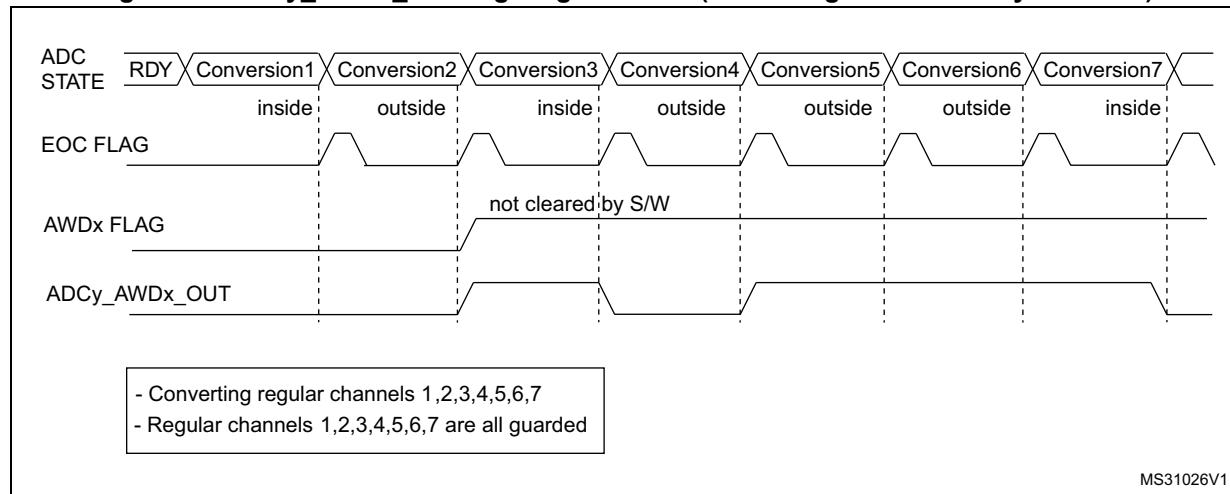


Figure 80. ADCy\_AWDx\_OUT signal generation (on a single regular channel)

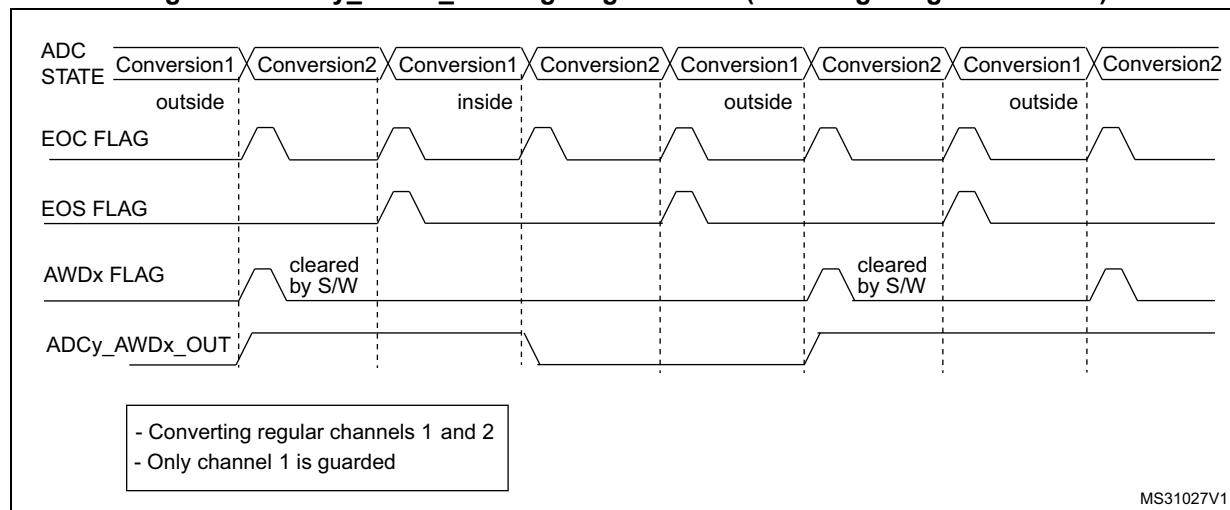
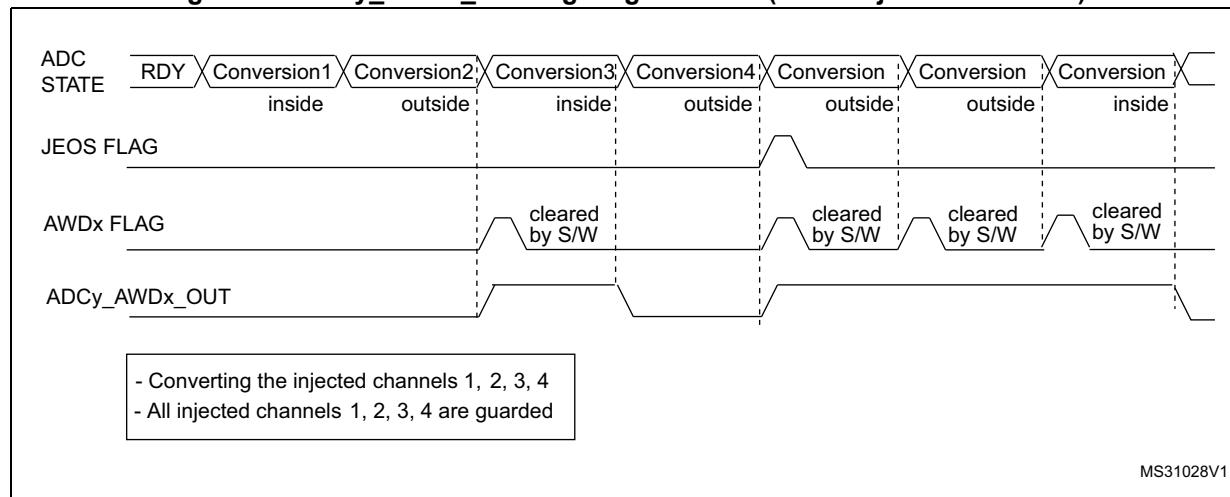


Figure 81. ADCy\_AWDx\_OUT signal generation (on all injected channels)



### 16.4.29 Oversampler

The oversampling unit performs data pre-processing to offload the CPU. It is able to handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

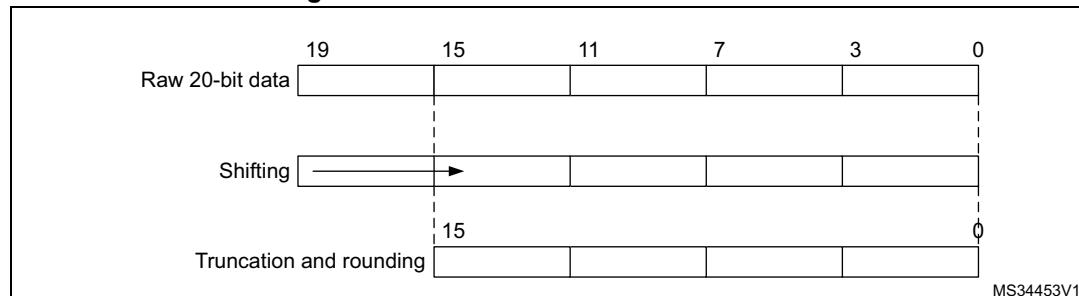
It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADC\_CFGR2 register, and can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits, and is defined using the OVSS[3:0] bits in the ADC\_CFGR2 register.

The summation unit can yield a result up to 20 bits (256x 12-bit results), which is first shifted right. It is then truncated to the 16 least significant bits, rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC\_DR data register.

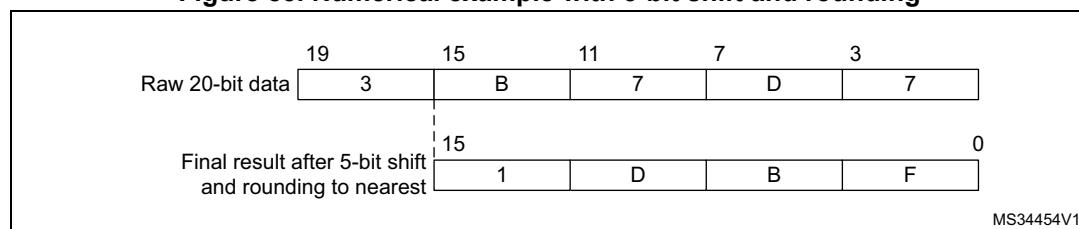
**Note:** *If the intermediary result after the shifting exceeds 16-bit, the result is truncated as is, without saturation.*

**Figure 82. 20-bit to 16-bit result truncation**



**Figure 83** gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

**Figure 83. Numerical example with 5-bit shift and rounding**



**Table 86** gives the data format for the various N and M combinations, for a raw conversion data equal to 0xFFFF.

**Table 86. Maximum output results versus N and M (gray cells indicate truncation)**

Over sampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS = 0111	8-bit shift OVSS = 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFFF0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0x1FFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversions, with an equivalent delay equal to  $N \times T_{CONV} = N \times (t_{SMPL} + t_{SAR})$ . The flags are set as follows:

- The end of the sampling phase (EOSMP) is set after each sampling phase
- The end of conversion (EOC) occurs once every N conversions, when the oversampled result is available
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (i.e. after  $N \times$  sequence length conversions total)

### ADC operating modes supported when oversampling

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous mode conversions
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC\_CFGR1 register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

**Note:** The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC\_CFGR1 is ignored and the data are always provided right-aligned.

Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSETy\_EN bit in ADC\_OFRy register is ignored (considered as reset).

## Analog watchdog

The analog watchdog functionality is maintained, with the following difference:

- The RES[1:0] bits are ignored, comparison is always done using the full 12-bit values HT[11:0] and LT[11:0]
- the comparison is performed on the most significant 12-bit of the 16-bit oversampled results ADC\_DR[15:4]

### Note:

*Care must be taken when using high shifting values, since this reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits, thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC\_DR[11:4] and HT[0:7] / LT[0:7], and HT[11:8] / LT[11:8] must be kept reset.*

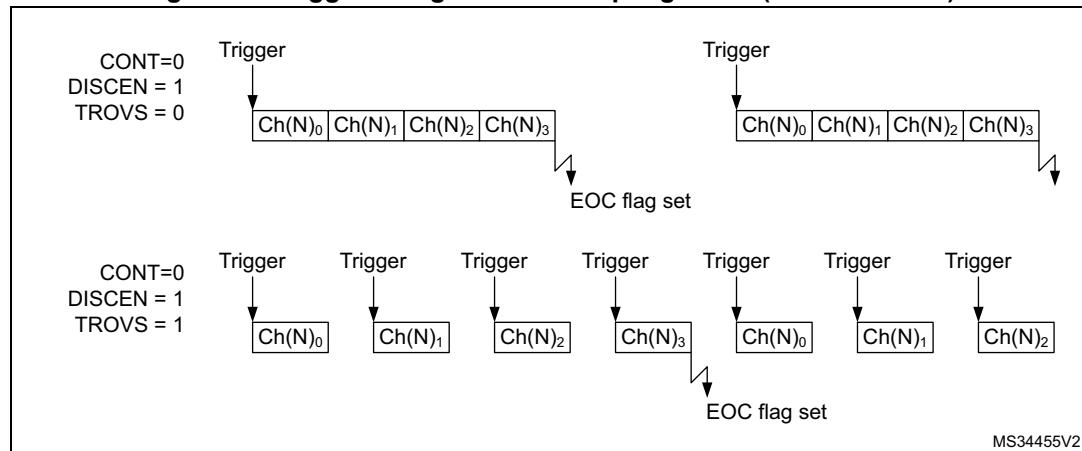
## Triggered mode

The averager can also be used for basic filtering purpose. Although not a very powerful filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TROVS bit in ADC\_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

[Figure 84](#) below shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

**Figure 84. Triggered regular oversampling mode (TROVS bit = 1)**



## Injected and regular sequencer management when oversampling

In oversampling mode, it is possible to have differentiated behavior for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

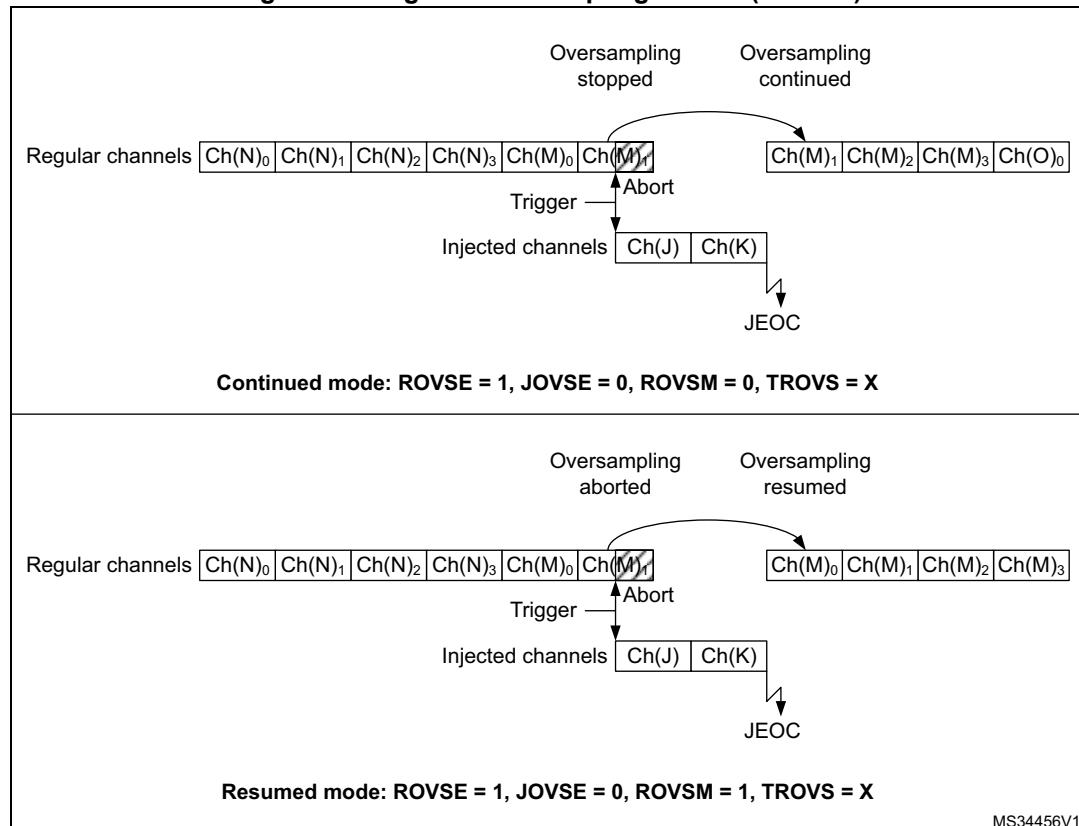
### Oversampling regular channels only

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- In continued mode, the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is complete whatever the injection frequency (providing at least one regular conversion can be complete between triggers);
- In resumed mode, the accumulation restarts from 0 (previous conversion results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be complete and the regular sequencer is blocked.

Figure 85 gives examples for a 4x oversampling ratio.

**Figure 85. Regular oversampling modes (4x ratio)**



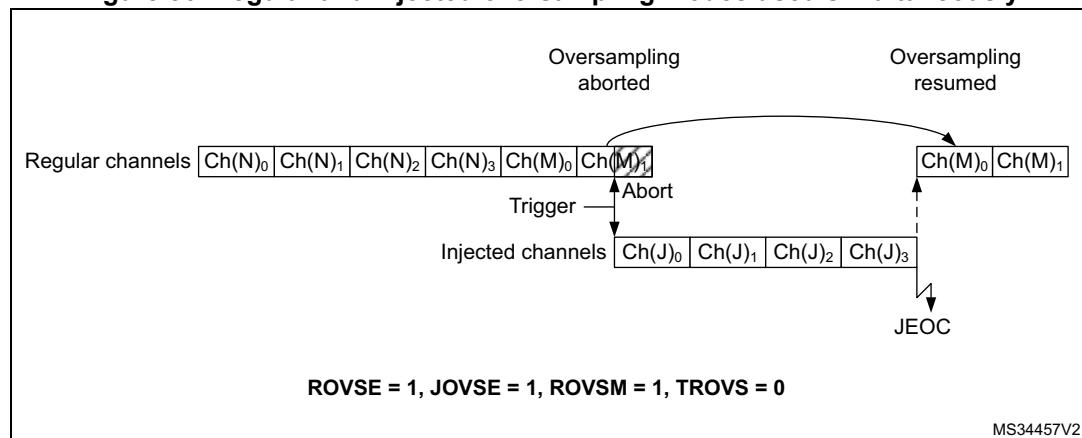
### Oversampling Injected channels only

The Injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

### Oversampling regular and Injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling mode is forced to resumed mode (ROVSM bit ignored), as represented on [Figure 86](#) below.

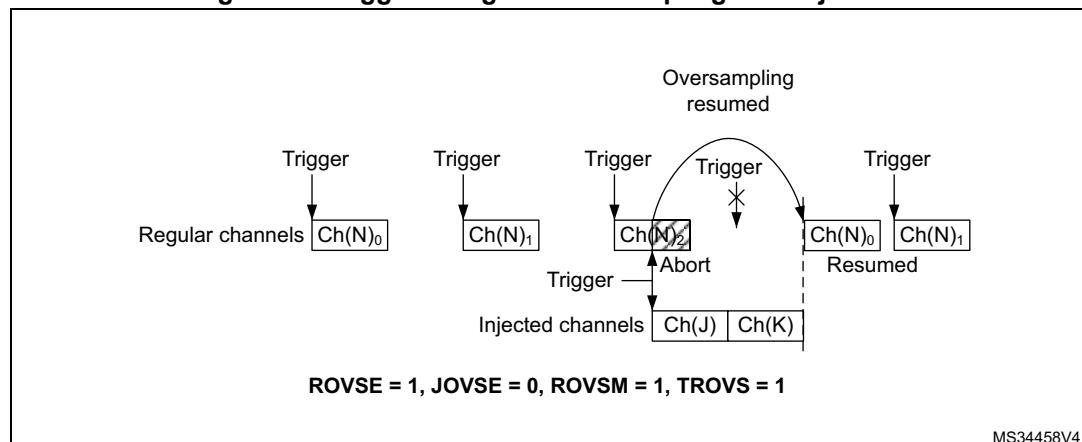
**Figure 86. Regular and injected oversampling modes used simultaneously**



### Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented on [Figure 87](#) below.

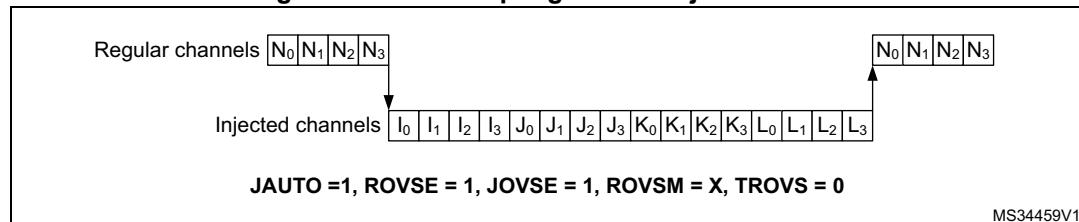
**Figure 87. Triggered regular oversampling with injection**



### Auto-injected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The [Figure 88](#) below shows how the conversions are sequenced.

**Figure 88. Oversampling in auto-injected mode**



It is possible to have also the triggered mode enabled, using the TROVS bit. In this case, the ADC must be configured as following: JAUTO = 1, DISCEN = 0, JDISCEN = 0, ROVSE = 1, JOVSE = 1 and TROVS = 1.

#### 16.4.30 Temperature sensor

The temperature sensor can be used to measure the junction temperature ( $T_j$ ) of the device. The temperature sensor is internally connected to the ADC input channels which are used to convert the sensor output voltage to a digital value. When not in use, the sensor can be put in power down mode. It support the temperature range  $-40$  to  $125$   $^{\circ}\text{C}$ .

[Figure 89](#) shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to  $45$   $^{\circ}\text{C}$  from one chip to another).

The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

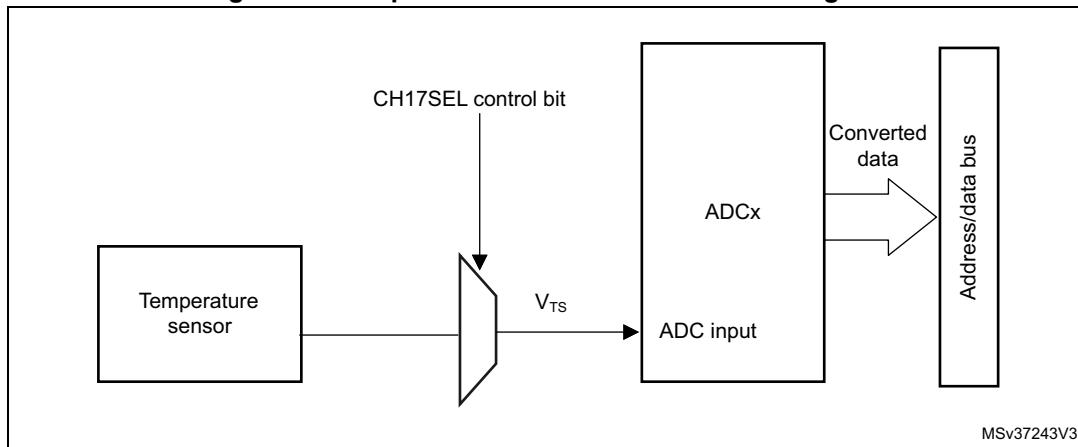
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet for additional information).

The temperature sensor is internally connected to the ADC input channel which is used to convert the sensor's output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

[Figure 89](#) shows the block diagram of the temperature sensor.

Figure 89. Temperature sensor channel block diagram



### Reading the temperature

To use the sensor:

1. Select the ADC input channels that is connected to V<sub>TS</sub>.
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the CH17SEL bit in the ADCx\_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V<sub>TS</sub> data in the ADC data register.
6. Calculate the actual temperature using the following formula:

$$\text{Temperature (in } ^\circ\text{C)} = \frac{\text{TS\_CAL2\_TEMP} - \text{TS\_CAL1\_TEMP}}{\text{TS\_CAL2} - \text{TS\_CAL1}} \times (\text{TS\_DATA} - \text{TS\_CAL1}) + \text{TS\_CAL1\_TEMP}$$

Where:

- TS\_CAL2 is the temperature sensor calibration value acquired at TS\_CAL2\_TEMP.
- TS\_CAL1 is the temperature sensor calibration value acquired at TS\_CAL1\_TEMP.
- TS\_DATA is the actual temperature sensor output value converted by ADC.

Refer to the device datasheet for more information about TS\_CAL1 and TS\_CAL2 calibration points.

**Note:** The sensor has a startup time after waking from power-down mode before it can output V<sub>TS</sub> at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and CH17SEL bits should be set at the same time.

The above formula is given for TS\_DATA measurement done with the same V<sub>REF+</sub> voltage as TS\_CAL1/TS\_CAL2 values. If V<sub>REF+</sub> is different, the formula must be adapted. For example if V<sub>REF+</sub> = 3.3 V and TS\_CAL data are acquired at V<sub>REF+</sub> = 3.0 V, TS\_DATA must be replaced by TS\_DATA x (3.3/3.0).

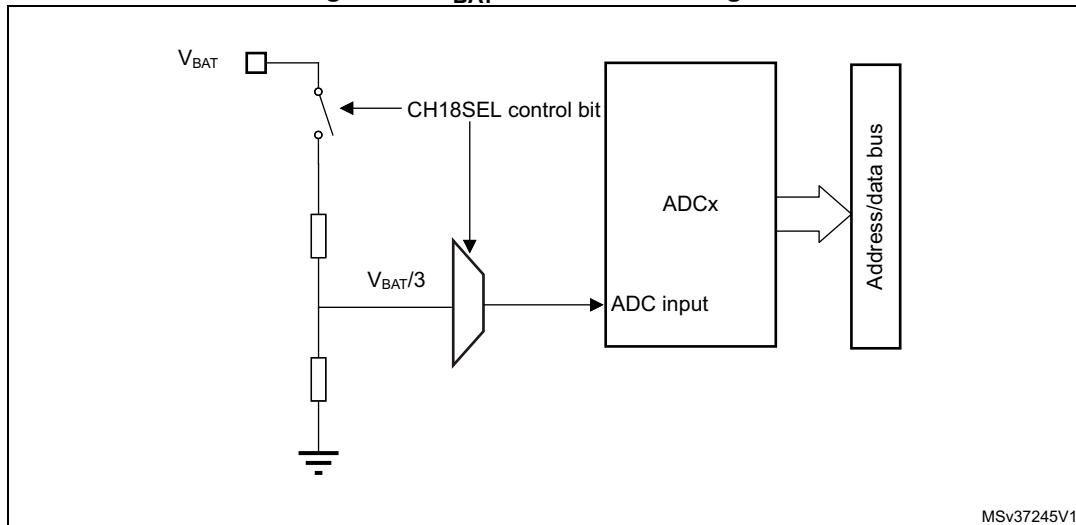
### 16.4.31 $V_{BAT}$ supply monitoring

The CH18SEL bit in the ADCx\_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 3. This bridge is automatically enabled when CH18SEL is set, to connect  $V_{BAT}/3$  to the ADC input channels. As a consequence, the converted digital value is one third of the  $V_{BAT}$  voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the  $V_{BAT}/3$  voltage.

The figure below shows the block diagram of the  $V_{BAT}$  sensing feature.

Figure 90.  $V_{BAT}$  channel block diagram



1. The CH18SEL bit must be set to enable the conversion of internal channel for  $V_{BAT}/3$ .

### 16.4.32 Monitoring the internal voltage reference

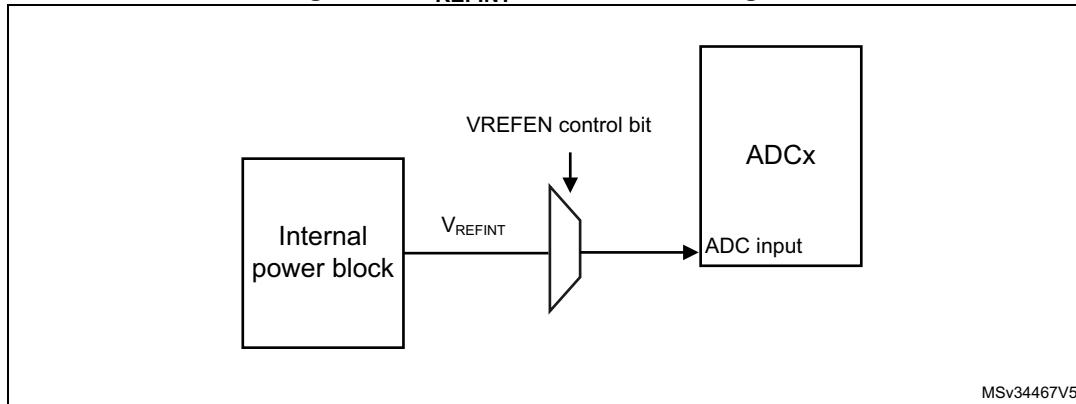
It is possible to monitor the internal voltage reference ( $V_{REFINT}$ ) to have a reference point for evaluating the ADC  $V_{REF+}$  voltage level.

The internal voltage reference is internally connected to the input channel 0 of the ADC1 (ADC1\_INP0).

Refer to the electrical characteristics section of the product datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

*Figure 91* shows the block diagram of the  $V_{REFINT}$  sensing feature.

**Figure 91.  $V_{REFINT}$  channel block diagram**



1. The VREFEN bit into ADCx\_CCR register must be set to enable the conversion of internal channels ( $V_{REFINT}$ ).

## Calculating the actual $V_{REF+}$ voltage using the internal reference voltage

The power supply voltage applied to the device may be subject to variations or not precisely known. When  $V_{DDA}$  is connected to  $V_{REF+}$ , it is possible to compute the actual  $V_{DDA}$  voltage using the embedded internal reference voltage ( $V_{REFINT}$ ).  $V_{REFINT}$  and its calibration data, acquired by the ADC during the manufacturing process at  $V_{DDA\_Charac}$ , can be used to evaluate the actual  $V_{DDA}$  voltage level.

The following formula gives the actual  $V_{REF+}$  voltage supplying the device:

$$V_{REF+} = V_{REF+ \text{ Charac}} \times VREFINT\_CAL / VREFINT\_DATA$$

Where:

- $V_{REF+\_Charac}$  is the value of  $V_{REF+}$  voltage characterized at  $V_{REFINT}$  during the manufacturing process. It is specified in the device datasheet.
  - $V_{REFINT\_CAL}$  is the  $V_{REFINT}$  calibration value
  - $V_{REFINT\_DATA}$  is the actual  $V_{REFINT}$  output value converted by ADC

## Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between  $V_{REF+}$  and the voltage applied on the converted channel.

For most applications  $V_{DDA}$  value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from  $V_{DDA}$ :

$$V_{\text{CHANNEL}x} = \frac{V_{\text{REF+}}}{\text{FULL SCALE}} \times \text{ADC\_DATA}$$

By replacing  $V_{REF+}$  by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{V_{REF+\_Charac} \times VREFINT\_CAL \times ADC\_DATA}{VREFINT\_DATA \times FULL\_SCALE}$$

For applications where  $V_{REF+}$  is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL\_SCALE} \times ADC\_DATA$$

Where:

- $V_{REF+\_Charac}$  is the value of  $V_{REF+}$  voltage characterized at  $V_{REFINT}$  during the manufacturing process.
- $VREFINT\_CAL$  is the  $V_{REFINT}$  calibration value
- $ADC\_DATA$  is the value measured by the ADC on channel x (right-aligned)
- $VREFINT\_DATA$  is the actual  $V_{REFINT}$  output value converted by the ADC
- $FULL\_SCALE$  is the maximum digital value of the ADC output. For example with 12-bit resolution, it is  $2^{12} - 1 = 4095$  or with 8-bit resolution,  $2^8 - 1 = 255$ .

*Note:* If ADC measurements are done using an output format other than 16-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

## 16.5 ADC in low-power mode

Table 87. Effect of low-power modes on the ADC

Mode	Description
Sleep	No effect. DMA requests are functional.
Low-power run	No effect.
Low-power sleep	No effect. DMA requests are functional.
Stop 0/Stop 1	The ADC is not operational. Its state is kept. The ADC consumes the static current recommended to disable the peripheral in advance in order to reduce power consumption.
Stop 2	The ADC is not operational. Its state is kept. The ADC regulator is disabled by hardware. ADC and regulator must be disabled before entering Stop 2 mode.
Standby	The ADC is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

## 16.6 ADC interrupts

An interrupt can be generated:

- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (flag OVR)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

**Table 88. ADC interrupts**

Interrupt event	Event flag	Enable control bit
ADC ready	ADRDY	ADRDYIE
End of conversion of a regular group	EOC	EOCIE
End of sequence of conversions of a regular group	EOS	EOSIE
End of conversion of a injected group	JEOC	JEOCIE
End of sequence of conversions of an injected group	JEOS	JEOSIE
Analog watchdog 1 status bit is set	AWD1	AWD1IE
Analog watchdog 2 status bit is set	AWD2	AWD2IE
Analog watchdog 3 status bit is set	AWD3	AWD3IE
End of sampling phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE
Injected context queue overflows	JQOVF	JQOVFIE

## 16.7 ADC registers

Refer to [Section 1.2 on page 61](#) for a list of abbreviations used in register descriptions.

### 16.7.1 ADC interrupt and status register (ADC\_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
					rc_w1										

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

#### Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to [Section 16.4.21: Queue of context for injected conversions](#) for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

#### Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC\_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

#### Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC\_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

#### Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC\_TR1 register. It is cleared by software writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

#### Bit 6 **JEOS**: Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

**Bit 5 JE0C: Injected channel end of conversion flag**

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC\_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC\_JDRy register

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)  
1: Injected channel conversion complete

**Bit 4 OVR: ADC overrun**

This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)  
1: Overrun has occurred

**Bit 3 EOS: End of regular sequence flag**

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)  
1: Regular Conversions sequence complete

**Bit 2 EOC: End of conversion flag**

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC\_DR register. It is cleared by software writing 1 to it or by reading the ADC\_DR register

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)  
1: Regular channel conversion complete

**Bit 1 EOSMP: End of sampling flag**

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)  
1: End of sampling phase reached

**Bit 0 ADRDY: ADC ready**

This bit is set by hardware after the ADC has been enabled (bit ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)  
1: ADC is ready to start conversion

### 16.7.2 ADC interrupt enable register (ADC\_IER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF IE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMP IE	ADRDY IE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.

0: Injected Context Queue Overflow interrupt disabled

1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

**Bit 5 JEOKIE:** End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.  
0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

**Bit 4 OVRIE:** Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.  
0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

**Bit 3 EOSIE:** End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.  
0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

**Bit 2 EOCIE:** End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.  
0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

**Bit 1 EOSMPIE:** End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.  
0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

**Bit 0 ADRDYIE:** ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 16.7.3 ADC control register (ADC\_CR)

Address offset: 0x08

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.						
rs	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

*Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0.*

*The software is allowed to update the calibration factor by writing ADC\_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing)*

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential inputs mode for the calibration.

0: Writing ADCAL launches a calibration in single-ended inputs mode.

1: Writing ADCAL launches a calibration in differential inputs mode.

*Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

*Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bits is set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC Voltage regulator disabled

1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to [Section 16.4.6: ADC Deep-power-down mode \(DEEPPWD\) and ADC voltage regulator \(ADVREGEN\)](#).

The software can program this bit field only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

*Note: The software is allowed to set JADSTP only when JADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC)*

*In Auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)*

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

*Note: The software is allowed to set ADSTP only when ADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC).*

*In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).*

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN[1:0], a conversion starts immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode when software trigger is selected (JEXTSEL = 0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.
- in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

*Note: The software is allowed to set JADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).*

*In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)*

**Bit 2 ADSTART:** ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTN[1:0], a conversion starts immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode when software trigger is selected (EXTSEL = 0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.
- in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

*Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC)*

*In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)*

**Bit 1 ADDIS:** ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

*Note: The software is allowed to set ADDIS only when ADEN = 1 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)*

**Bit 0 ADEN:** ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

*Note: The software is allowed to set ADEN only when all bits of ADC\_CR registers are 0 (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)*

### 16.7.4 ADC configuration register (ADC\_CFGR)

Address offset: 0x0C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JQDIS	AWD1CH[4:0]				JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	ALIGN	RES[1:0]		Res.	DMACFG	DMAEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **JQDIS**: Injected Queue disable

These bits are set and cleared by software to disable the Injected Queue mechanism :

0: Injected Queue enabled

1: Injected Queue disabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).*

*A set or reset of JQDIS bit causes the injected queue to be flushed and the JSQR register is cleared.*

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel 0 monitored by AWD1

00001: ADC analog input channel 1 monitored by AWD1

.....

10010: ADC analog input channel 18 monitored by AWD1

others: reserved, must not be used

*Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.*

*The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.*

*The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 25 **JAUTO**: Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).*

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 21 **JQM**: JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: JSQR mode 0: The Queue is never empty and maintains the last written configuration into JSQR.

1: JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to [Section 16.4.21: Queue of context for injected conversions](#) for more information.

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 20 **JDISCEN**: Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

*It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.*

Bits 19:17 **DISCNUM[2:0]**: Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable Discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

*Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.*

*It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.*

*The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

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

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

0: Auto-delayed conversion mode off

1: Auto-delayed conversion mode on

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

*Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.*

*The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC\_DR register is preserved with the old data when an overrun is detected.

1: ADC\_DR register is overwritten with the last conversion result when an overrun is detected.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bits 9:6 **EXTSEL[3:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

- 0000: Event 0
- 0001: Event 1
- 0010: Event 2
- 0011: Event 3
- 0100: Event 4
- 0101: Event 5
- 0110: Event 6
- 0111: Event 7

...

- 1111: Event 15

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 5 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to [Section : Data register, data alignment and offset \(ADC\\_DR, OFFSETy, OFFSETy\\_CH, ALIGN\)](#)

- 0: Right alignment
- 1: Left alignment

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 4:3 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

- 00: 12-bit
- 01: 10-bit
- 10: 8-bit
- 11: 6-bit

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMACFG**: Direct memory access configuration

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.

- 0: DMA One Shot mode selected
- 1: DMA Circular mode selected

For more details, refer to [Section : Managing conversions using the DMA](#)

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA to manage automatically the converted data. For more details, refer to [Section : Managing conversions using the DMA](#).

- 0: DMA disabled
- 1: DMA enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 16.7.5 ADC configuration register 2 (ADC\_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

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

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

Bits 27:17 Reserved, must be kept at reset value.

Bits 16:11 Reserved, must be kept at reset value.

#### Bit 10 ROVSM: Regular Oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injection sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injection sequence start)

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

#### Bit 9 TROVS: Triggered Regular Oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

#### Bits 8:5 OVSS[3:0]: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift

0001: Shift 1-bit

0010: Shift 2-bits

0011: Shift 3-bits

0100: Shift 4-bits

0101: Shift 5-bits

0110: Shift 6-bits

0111: Shift 7-bits

1000: Shift 8-bits

Other codes reserved

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 4:2 **OVSR[2:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

000: 2x

001: 4x

010: 8x

011: 16x

100: 32x

101: 64x

110: 128x

111: 256x

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 1 **JOVSE**: Injected Oversampling Enable

This bit is set and cleared by software to enable injected oversampling.

0: Injected Oversampling disabled

1: Injected Oversampling enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)*

Bit 0 **ROVSE**: Regular Oversampling Enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular Oversampling disabled

1: Regular Oversampling enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)*

### 16.7.6 ADC sample time register 1 (ADC\_SMPR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		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 (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

- 000: 2.5 ADC clock cycles
- 001: 6.5 ADC clock cycles
- 010: 12.5 ADC clock cycles
- 011: 24.5 ADC clock cycles
- 100: 47.5 ADC clock cycles
- 101: 92.5 ADC clock cycles
- 110: 247.5 ADC clock cycles
- 111: 640.5 ADC clock cycles

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.*

### 16.7.7 ADC sample time register 2 (ADC\_SMPR2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 26:0 **SMPx[2:0]**: Channel x sampling time selection (x = 18 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

- 000: 2.5 ADC clock cycles
- 001: 6.5 ADC clock cycles
- 010: 12.5 ADC clock cycles
- 011: 24.5 ADC clock cycles
- 100: 47.5 ADC clock cycles
- 101: 92.5 ADC clock cycles
- 110: 247.5 ADC clock cycles
- 111: 640.5 ADC clock cycles

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.*

### 16.7.8 ADC watchdog threshold register 1 (ADC\_TR1)

Address offset: 0x20

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.	Res.	Res.	Res.	HT1[11:0]													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Res.	Res.	Res.	Res.	LT1[11:0]													
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

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

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1.

Refer to [Section 16.4.28: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWDHTx, AWD\\_LTx, AWDX\)](#)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

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

Bits 11:0 **LT1[11:0]**: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 1.

Refer to [Section 16.4.28: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWDHTx, AWD\\_LTx, AWDX\)](#)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 16.7.9 ADC watchdog threshold register 2 (ADC\_TR2)

Address offset: 0x24

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16						
Res.	HT2[7:0]																				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Res.	LT2[7:0]																				
								rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT2[7:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 2.

Refer to [Section 16.4.28: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD\\_HTx, AWD\\_LTx, AWDx\)](#)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

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

Bits 7:0 **LT2[7:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 2.

Refer to [Section 16.4.28: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD\\_HTx, AWD\\_LTx, AWDx\)](#)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 16.7.10 ADC watchdog threshold register 3 (ADC\_TR3)

Address offset: 0x28

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16						
Res.	HT3[7:0]																				
								rw	rw	rw	rw	rw	rw	rw	rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Res.	LT3[7:0]																				
								rw	rw	rw	rw	rw	rw	rw	rw						

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT3[7:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to [Section 16.4.28: Analog window watchdog \(AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD\\_HTx, AWD\\_LTx, AWDx\)](#)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

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

Bits 7:0 **LT3[7:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3.

This watchdog compares the 8-bit of LT3 with the 8 MSB of the converted data.

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 16.7.11 ADC regular sequence register 1 (ADC\_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]				Res.	SQ3[4:0]				Res.	SQ2[4]		
			rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]				Res.	Res.	L[3:0]				
rw	rw	rw	rw		rw	rw	rw	rw			rw	rw	rw	rw	

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]**: 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 4th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]**: 3rd conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]**: 2nd conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]**: 1st conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 1st in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]**: Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

0000: 1 conversion

0001: 2 conversions

...

1111: 16 conversions

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

*Note: Some channels are not connected physically and must not be selected for conversion.*

### 16.7.12 ADC regular sequence register 2 (ADC\_SQR2)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]				Res.	SQ8[4:0]				Res.	SQ7[4]		
			rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]				Res.	SQ5[4:0]					
rw	rw	rw	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]**: 9th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 9th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]**: 8th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 8th in the regular conversion sequence

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]**: 7th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 7th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]**: 6th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 6th in the regular conversion sequence

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]**: 5th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 5th in the regular conversion sequence

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

*Note: Some channels are not connected physically and must not be selected for conversion.*

### 16.7.13 ADC regular sequence register 3 (ADC\_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]**: 14th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 14th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]**: 13th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 13th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]**: 12th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 12th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]**: 11th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 11th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]**: 10th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 10th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

*Note: Some channels are not connected physically and must not be selected for conversion.*

### 16.7.14 ADC regular sequence register 4 (ADC\_SQR4)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	Res.	Res.	Res.	Res.			SQ16[4:0]		Res.		SQ15[4:0]		rw	rw	rw	rw
					rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

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

Bits 10:6 **SQ16[4:0]**: 16th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 16th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]**: 15th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 15th in the regular conversion sequence.

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

*Note: Some channels are not connected physically and must not be selected for conversion.*

### 16.7.15 ADC regular data register (ADC\_DR)

Address offset: 0x40

Reset value: 0x0000 0000

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

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

Bits 15:0 **RDATA[15:0]**: Regular data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in [Section 16.4.26: Data management](#).

### 16.7.16 ADC injected sequence register (ADC\_JSQR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:2]		
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[1:0]		Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[3:0]				JL[1:0]	
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **JSQ4[4:0]**: 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 4th in the injected conversion sequence.

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **JSQ3[4:0]**: 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the injected conversion sequence.

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 19 Reserved, must be kept at reset value.

Bits 18:14 **JSQ2[4:0]**: 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the injected conversion sequence.

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 13 Reserved, must be kept at reset value.

Bits 12:8 **JSQ1[4:0]**: 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 1st in the injected conversion sequence.

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bits 7:6 **JEXTEN[1:0]**: External Trigger Enable and Polarity Selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: If JQDIS = 0 (queue enabled), Hardware and software trigger detection disabled  
00: If JQDIS = 1 (queue disabled), Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

*If JQM = 1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 16.4.21: Queue of context for injected conversions)*

Bits 5:2 **JEXTSEL[3:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

0000: Event 0

0001: Event 1

0010: Event 2

0011: Event 3

0100: Event 4

0101: Event 5

0110: Event 6

0111: Event 7

...

1111: Event 15

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bits 1:0 **JL[1:0]**: Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

*Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

*Note: Some channels are not connected physically and must not be selected for conversion.*

### 16.7.17 ADC offset y register (ADC\_OFRy)

Address offset: 0x60 + 0x04 \* (y -1), (y= 1 to 4)

Reset value: 0x0000 0000

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

Bit 31 **OFFSET\_EN**: Offset y enable

This bit is written by software to enable or disable the offset programmed into bits OFFSETy[11:0].

*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 30:26 **OFFSET\_CH[4:0]**: Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSETy[11:0] applies.

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*Some channels are not connected physically and must not be selected for the data offset y.*

Bits 25:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSET[11:0]**: Data offset y for the channel programmed into bits OFFSETy\_CH[4:0]

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the bits OFFSETy\_CH[4:0]. The conversion result can be read from in the ADC\_DR (regular conversion) or from in the ADC\_JDRy registers (injected conversion).

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*If several offset (OFFSETy) point to the same channel, only the offset with the lowest x value is considered for the subtraction.*

*Ex: if OFFSET1\_CH[4:0]=4 and OFFSET2\_CH[4:0]=4, this is OFFSET1[11:0] which is subtracted when converting channel 4.*

### 16.7.18 ADC injected channel y data register (ADC\_JDRy)

Address offset:  $0x80 + 0x04 * (y - 1)$ , ( $y = 1$  to  $4$ )

Reset value: 0x0000 0000

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

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

Bits 15:0 **JDATA[15:0]**: Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in [Section 16.4.26: Data management](#).

### 16.7.19 ADC analog watchdog 2 configuration register (ADC\_AWD2CR)

Address offset: 0xA0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2CH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 18:0 **AWD2CH[18:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWD2CH[i] = 0: ADC analog input channel i is not monitored by AWD2

AWD2CH[i] = 1: ADC analog input channel i is monitored by AWD2

When AWD2CH[18:0] = 000..0, the analog watchdog 2 is disabled

*Note: The channels selected by AWD2CH must be also selected into the SQRI or JSQRI registers.*

*The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*Some channels are not connected physically and must not be selected for the analog watchdog.*

### 16.7.20 ADC analog watchdog 3 configuration register (ADC\_AWD3CR)

Address offset: 0xA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[18:16]		
														rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
AWD3CH[15:0]																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

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

Bits 18:0 **AWD3CH[18:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWD3CH[i] = 0: ADC analog input channel i is not monitored by AWD3

AWD3CH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWD3CH[18:0] = 000..0, the analog watchdog 3 is disabled

*Note: The channels selected by AWD3CH must be also selected into the SQR*i* or JSQR*i* registers.*

*The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

*Some channels are not connected physically and must not be selected for the analog watchdog.*

### 16.7.21 ADC differential mode selection register (ADC\_DIFSEL)

Address offset: 0xB0

Reset value: 0x0000 0000

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

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

Bits 18:0 **DIFSEL[18:0]**: Differential mode for channels 18 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or differential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in single ended mode

DIFSEL[i] = 1: ADC analog input channel i is configured in differential mode

*Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).*

*The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

### 16.7.22 ADC calibration factors (ADC\_CALFACT)

Address offset: 0xB4

Reset value: 0x0000 0000

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

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

Bits 22:16 **CALFACT\_D[6:0]**: Calibration Factors in differential mode

These bits are written by hardware or by software.

Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new differential calibration is launched.

*Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).*

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

Bits 6:0 **CALFACT\_S[6:0]**: Calibration Factors In single-ended mode

These bits are written by hardware or by software.

Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new single-ended calibration is launched.

*Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).*

## 16.8 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

### 16.8.1 ADC common status register (ADC\_CSR)

Address offset: 0x300

Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADC\_ISR register.

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

Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
					r	r	r	r	r	r	r	r	r	r	r

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

Bit 10 **JQOVF\_MST**: Injected Context Queue Overflow flag of the master ADC  
This bit is a copy of the JQOVF bit in the corresponding ADC\_ISR register.

Bit 9 **AWD3\_MST**: Analog watchdog 3 flag of the master ADC  
This bit is a copy of the AWD3 bit in the corresponding ADC\_ISR register.

Bit 8 **AWD2\_MST**: Analog watchdog 2 flag of the master ADC  
This bit is a copy of the AWD2 bit in the corresponding ADC\_ISR register.

Bit 7 **AWD1\_MST**: Analog watchdog 1 flag of the master ADC  
This bit is a copy of the AWD1 bit in the corresponding ADC\_ISR register.

Bit 6 **JEOS\_MST**: End of injected sequence flag of the master ADC  
This bit is a copy of the JEOS bit in the corresponding ADC\_ISR register.

Bit 5 **JEOC\_MST**: End of injected conversion flag of the master ADC  
This bit is a copy of the JEOC bit in the corresponding ADC\_ISR register.

Bit 4 **OVR\_MST**: Overrun flag of the master ADC  
This bit is a copy of the OVR bit in the corresponding ADC\_ISR register.

Bit 3 **EOS\_MST**: End of regular sequence flag of the master ADC  
This bit is a copy of the EOS bit in the corresponding ADC\_ISR register.

Bit 2 **EOC\_MST**: End of regular conversion of the master ADC  
This bit is a copy of the EOC bit in the corresponding ADC\_ISR register.

Bit 1 **EOSMP\_MST**: End of Sampling phase flag of the master ADC  
This bit is a copy of the EOSMP bit in the corresponding ADC\_ISR register.

Bit 0 **ADRDY\_MST**: Master ADC ready  
This bit is a copy of the ADRDY bit in the corresponding ADC\_ISR register.

## 16.8.2 ADC common control register (ADC\_CCR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CH18SEL	CH17SEL	VREFEN	PRESC[3:0]				CKMODE[1:0]							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Res.															

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

Bit 24 **CH18SEL**: CH18 selection

This bit is set and cleared by software to control channel 18.

0:  $V_{BAT}$  channel disabled.

1:  $V_{BAT}$  channel enabled

Bit 23 **CH17SEL**: CH17 selection

This bit is set and cleared by software to control channel 17.

0: Temperature sensor channel disabled

1: Temperature sensor channel enabled

Bit 22 **VREFEN**:  $V_{REFINT}$  enable

This bit is set and cleared by software to enable/disable the  $V_{REFINT}$  channel.

0:  $V_{REFINT}$  channel disabled

1:  $V_{REFINT}$  channel enabled

Bits 21:18 **PRESC[3:0]**: ADC prescaler

These bits are set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

0000: input ADC clock not divided

0001: input ADC clock divided by 2

0010: input ADC clock divided by 4

0011: input ADC clock divided by 6

0100: input ADC clock divided by 8

0101: input ADC clock divided by 10

0110: input ADC clock divided by 12

0111: input ADC clock divided by 16

1000: input ADC clock divided by 32

1001: input ADC clock divided by 64

1010: input ADC clock divided by 128

1011: input ADC clock divided by 256

other: reserved

*Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0). The ADC prescaler value is applied only when CKMODE[1:0] = 00.*

Bits 17:16 **CKMODE[1:0]**: ADC clock mode

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

00: CK\_ADCx (x = 123) (Asynchronous clock mode), generated at product level (refer to Section 6: Reset and clock control (RCC))

01: HCLK/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set (HPRE[3:0] = 0xxx in RCC\_CFGR register) and if the system clock has a 50% duty cycle.

10: HCLK/2 (Synchronous clock mode)

11: HCLK/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

*Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

Bits 15:0 Reserved, must be kept at reset value.

## 16.9 ADC register map

**Table 89. ADC register map and reset values**

**Table 89. ADC register map and reset values (continued)**

**Table 89. ADC register map and reset values (continued)**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0xB4	ADC_CALFACT	Res.																																
	Reset value																																	

**Table 90. ADC register map and reset values (master and slave ADC common registers) offset = 0x300**

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_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
		Reset value																																	
0x04	Reserved																																		
0x08	ADC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	Reset value																																		

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 17 Voltage reference buffer (VREFBUF)

### 17.1 Introduction

The devices embed a voltage reference buffer which can be used as voltage reference for ADC and also as voltage reference for external components through the VREF+ pin. When the VREF+ pin is double-bonded with VDDA pin in a package, the voltage reference buffer is not available and must be kept disabled (refer to datasheet for packages pinout description).

### 17.2 VREFBUF implementation

Table 91. VREFBUF implementation

Feature	STM32WB55xx	STM32WB35xx
VREFBUF	X	-

### 17.3 VREFBUF functional description

The internal voltage reference buffer supports two voltages<sup>(a)</sup>, which are configured with VRS bits in the VREFBUF\_CSR register:

- VRS = 0: V<sub>REF\_OUT1</sub> around 2.048 V.
- VRS = 1: V<sub>REF\_OUT2</sub> around 2.5 V.

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 92. VREF buffer modes

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer off mode: – V <sub>REF+</sub> pin pulled-down to V <sub>SSA</sub>
0	1	External voltage reference mode (default value): – VREFBUF buffer off – V <sub>REF+</sub> pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer on – V <sub>REF+</sub> pin connected to VREFBUF buffer output
1	1	Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state

a. The minimum V<sub>DDA</sub> voltage depends on VRS setting, refer to the product datasheet.

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF\_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

## 17.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. For the VRS = 1 setting, the calibration data is automatically loaded to the TRIM register at reset. For the VRS = 0 setting, the software must take care of copying the calibration data from the read-only system memory area (flash memory) to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits.

Table 93. VREFBUF trimming data

Calibration value name	Description	Memory address
VREF_SC0	VREFBUF trimming value for VRS = 0	0x1FFF 75F0
VREF_SC1	VREFBUF trimming value for VRS = 1	0x1FFF 7530

## 17.5 VREFBUF registers

### 17.5.1 VREFBUF control and status register (VREFBUF\_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

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

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

Bit 3 **VRR**: Voltage reference buffer ready

0: the voltage reference buffer output is not ready.

1: the voltage reference buffer output reached the requested level.

Bit 2 **VRS**: Voltage reference scale

This bit selects the value generated by the voltage reference buffer.

0: Voltage reference set to  $V_{REF\_OUT1}$  (around 2.048 V).

1: Voltage reference set to  $V_{REF\_OUT2}$  (around 2.5 V).

Bit 1 **HIZ**: High impedance mode

This bit controls the analog switch to connect or not the  $V_{REF+}$  pin.

0:  $V_{REF+}$  pin is internally connected to the voltage reference buffer output.

1:  $V_{REF+}$  pin is high impedance.

Refer to [Table 92: VREF buffer modes](#) for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR**: Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.

0: Internal voltage reference mode disable (external voltage reference mode).

1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

## 17.5.2 VREFBUF calibration control register (VREFBUF\_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

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

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

Bits 5:0 **TRIM[5:0]**: Trimming code

For VRS = 1 (2.5 V), these bits are automatically initialized after reset with the trimming value stored in the flash memory during the production test.

For VRS = 0 (2.048V), the software must take care of copying the calibration data from the read-only system memory area (flash memory) to the TRIM[5:0] bitfield. Writing into these bits allows the tuning of the internal reference buffer voltage.

*Note: If the user application performs the trimming, the trimming code should start from 000000 to 111111 in ascending order.*

## 17.5.3 VREFBUF register map

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

**Table 94. VREFBUF register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	VREFBUF_CSR	Res.	VRR	VRS	HIZ	ENVR																											
	Reset value	0	0	0	0	0	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 94. VREFBUF register map and reset values**

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

## 18 Comparator (COMP)

### 18.1 Introduction

The devices embed two ultra-low-power comparators COMP1, and COMP2.

The comparators can be used for a variety of functions including:

- Wake-up from low-power mode triggered by an analog signal,
- Analog signal conditioning,
- Cycle-by-cycle current control loop when combined with a PWM output from a timer.

### 18.2 COMP main features

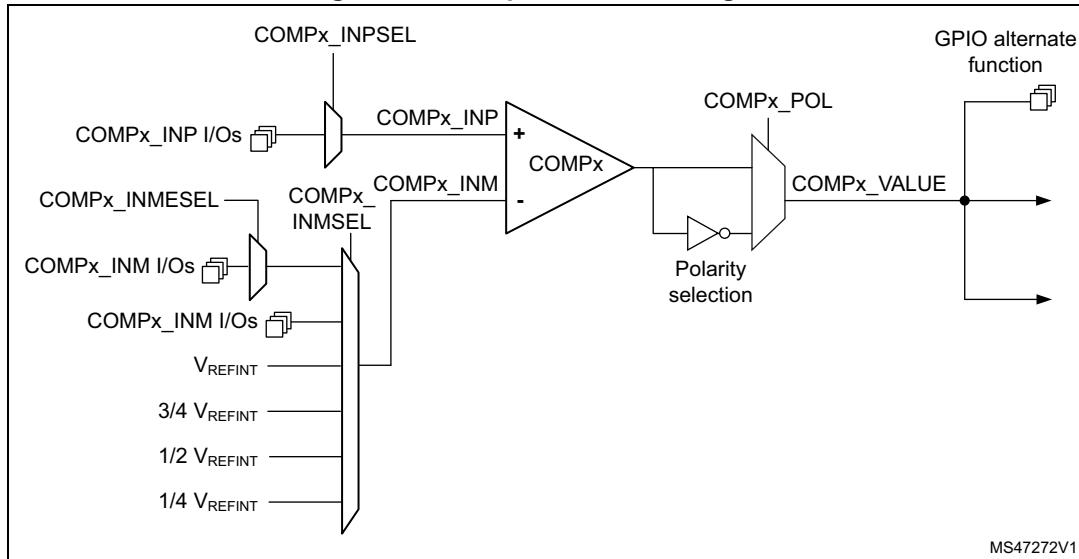
- Each comparator has configurable plus and minus inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- Programmable hysteresis
- Programmable speed / consumption
- The outputs can be redirected to an I/O or to timer inputs for triggering:
  - Break events for fast PWM shutdowns
- Comparator outputs with blanking source
- The two comparators can be combined in a window comparator
- Each comparator has interrupt generation capability with wake-up from Sleep and Stop modes (through the EXTI controller)

## 18.3 COMP functional description

### 18.3.1 COMP block diagram

The block diagram of the comparators is shown in [Figure 92](#).

**Figure 92. Comparator block diagram**



### 18.3.2 COMP pins and internal signals

The I/Os used as comparators inputs must be configured in analog mode in the GPIOs registers.

The comparator output can be connected to the I/Os using the alternate function channel given in “Alternate function mapping” table in the datasheet.

The output can also be internally redirected to a variety of timer input for the following purposes:

- Emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- Cycle-by-cycle current control, using OCREF\_CLR inputs
- Input capture for timing measures

It is possible to have the comparator output simultaneously redirected internally and externally.

**Table 95. COMP1 input plus assignment**

COMP1_INP	COMP1_INPSEL
PC5 (available only on STM32WB55xx)	00
PB2	01
PA1	10

Table 96. COMP1 input minus assignment

COMP1_INM	COMP1_INMSEL[2:0]	COMP1_INMESEL[1:0]
$\frac{1}{4} V_{REFINT}$	000	N. A. <sup>(1)</sup>
$\frac{1}{2} V_{REFINT}$	001	N. A. <sup>(1)</sup>
$\frac{3}{4} V_{REFINT}$	010	N. A. <sup>(1)</sup>
$V_{REFINT}$	011	N. A. <sup>(1)</sup>
Reserved	100	N. A. <sup>(1)</sup>
Reserved	101	N. A. <sup>(1)</sup>
PA9	110	N. A. <sup>(1)</sup>
PC4 (available only on STM32WB55xx)	111	00
PA0	111	01
PA4	111	10
PA5	111	11

1. N. A.: not affected.

Table 97. COMP2 input plus assignment

COMP2_INP	COMP2_INPSEL
PB4	00
PB6	01
PA3	10

Table 98. COMP2 input minus assignment

COMP2_INM	COMP2_INMSEL[2:0]	COMP2_INMESEL[1:0]
$\frac{1}{4} V_{REFINT}$	000	N.A. <sup>(1)</sup>
$\frac{1}{2} V_{REFINT}$	001	N.A. <sup>(1)</sup>
$\frac{3}{4} V_{REFINT}$	010	N.A. <sup>(1)</sup>
$V_{REFINT}$	011	N.A. <sup>(1)</sup>
Reserved	100	N.A. <sup>(1)</sup>
Reserved	101	N.A. <sup>(1)</sup>
PB3	110	N.A. <sup>(1)</sup>
PB7	111	00
PA2	111	01
PA4	111	10
PA5	111	11

1. N. A.: not affected.

### 18.3.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the APB2 clock.

There is no clock enable control bit provided in the RCC controller. Reset and clock enable bits are common for COMP and SYSCFG.

**Important:** The polarity selection logic and the output redirection to the port works independently from the APB2 clock. This allows the comparator to work even in Stop mode.

### 18.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary to insure that the comparator programming cannot be altered in case of spurious register access or program counter corruption.

For this purpose, the comparator control and status registers can be write-protected (read-only).

Once the programming is completed, the COMPx LOCK bit can be set to 1. This causes the whole register to become read-only, including the COMPx LOCK bit.

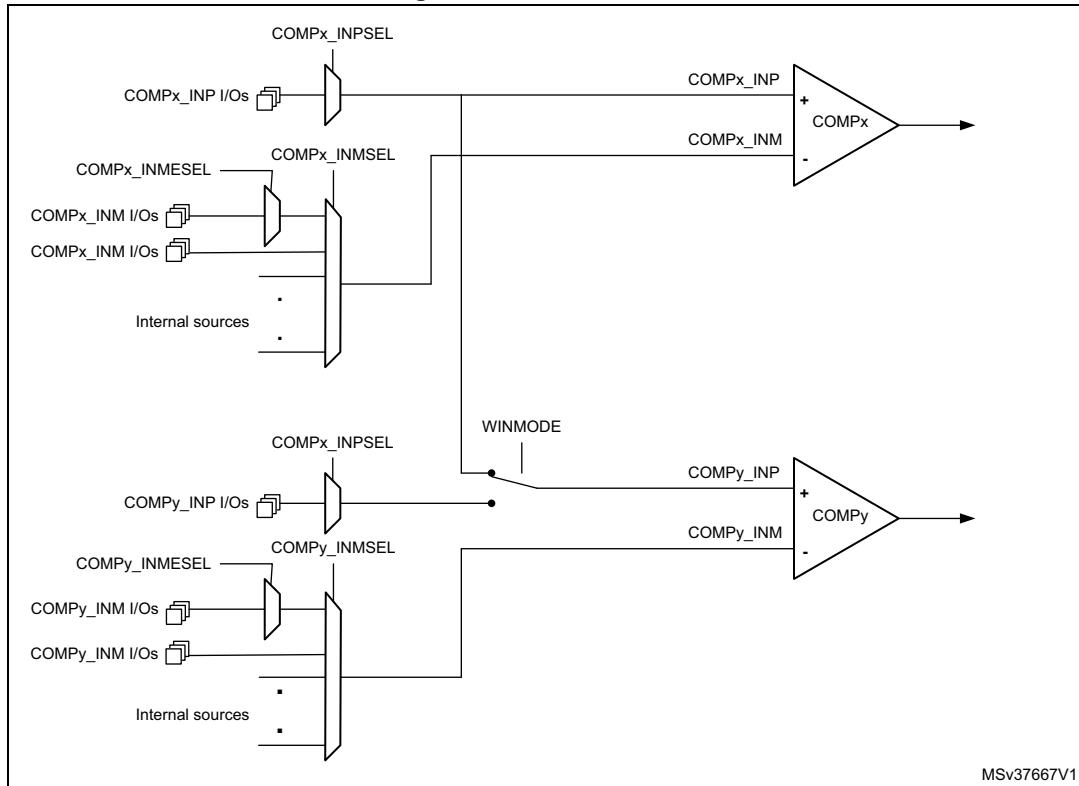
The write protection can only be reset by a MCU reset.

### 18.3.5 Window comparator

The purpose of window comparator is to monitor the analog voltage if it is within specified voltage range defined by lower and upper threshold.

Two embedded comparators can be utilized to create window comparator. The monitored analog voltage is connected to the non-inverting (plus) inputs of comparators connected together and the upper and lower threshold voltages are connected to the inverting (minus) inputs of the comparators. Two non-inverting inputs can be connected internally together by enabling WINMODE bit to save one IO for other purposes.

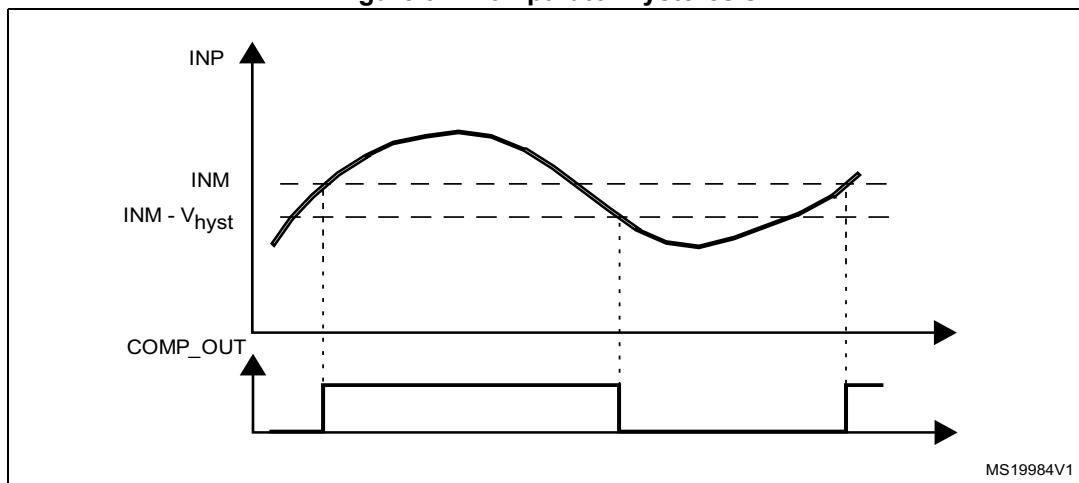
Figure 93. Window mode



### 18.3.6 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.

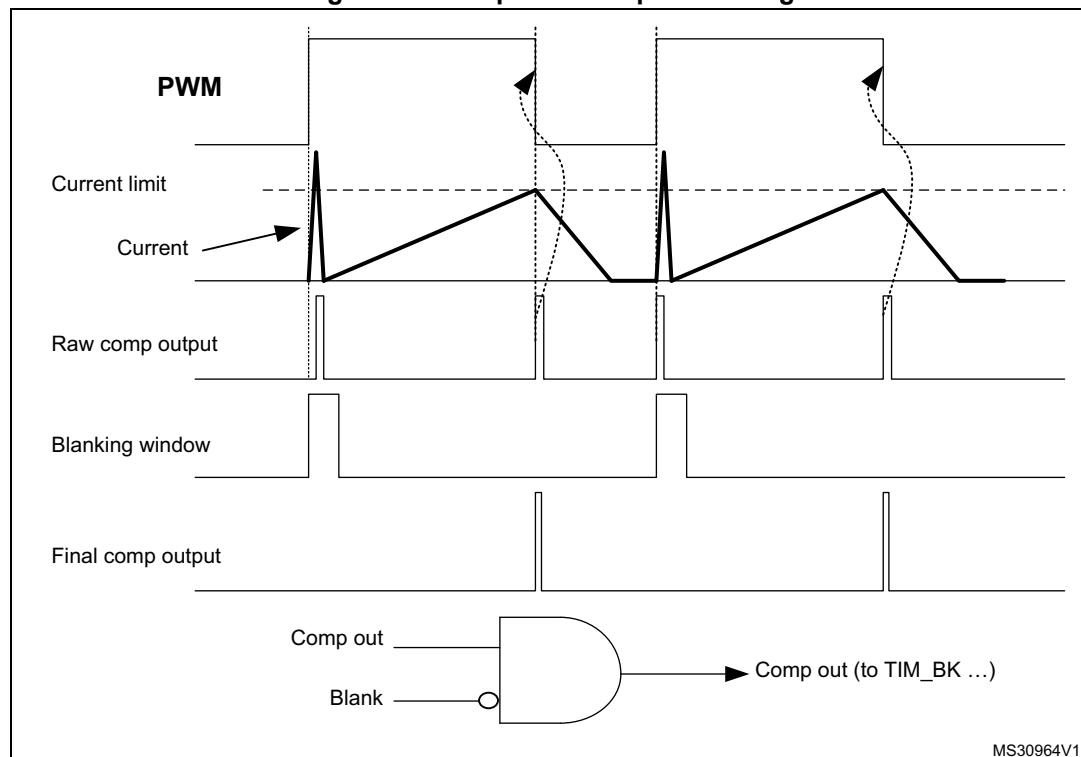
Figure 94. Comparator hysteresis



### 18.3.7 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes). It consists of a selection of a blanking window which is a timer output compare signal. The selection is done by software (refer to the comparator register description for possible blanking signals). Then, the complementary of the blanking signal is ANDed with the comparator output to provide the wanted comparator output. See the example provided in the figure below.

Figure 95. Comparator output blanking



### 18.3.8 COMP power and speed modes

COMP1 and COMP2 power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application.

The bits PWRMODE[1:0] in COMPx\_CSR registers can be programmed as follows:

- 00: High speed / full power
- 01 or 10: Medium speed / medium power
- 11: Low speed / ultra-low-power

## 18.4 COMP low-power modes

Table 99. Comparator behavior in the low power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.
Low-power run	No effect.
Low-power sleep	No effect. COMP interrupts cause the device to exit the Low-power sleep mode.
Stop 0	No effect on the comparators. Comparator interrupts cause the device to exit the Stop mode.
Stop 1	
Stop 2	
Standby	The COMP registers are powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

## 18.5 COMP interrupts

The comparator outputs are internally connected to the Extended interrupts and events controller. Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit from low-power modes.

Refer to Interrupt and events section for more details.

To enable COMPx interrupt, it is required to follow this sequence:

1. Configure and enable the EXTI line corresponding to the COMPx output event in interrupt mode and select the rising, falling or both edges sensitivity
2. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines
3. Enable COMPx.

Table 100. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit from Sleep mode	Exit from Stop modes	Exit from Standby mode
COMP1 output	VALUE in COMP1_CSR	Through EXTI	Yes	Yes	N/A
COMP2 output	VALUE in COMP2_CSR	Through EXTI	Yes	Yes	N/A

## 18.6 COMP registers

### 18.6.1 Comparator 1 control and status register (COMP1\_CSR)

The COMP1\_CSR is the Comparator 1 control/status register. It contains all the bits /flags related to comparator1.

Address offset: 0x00

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	INMESEL[1:0]	Res.	SCALEN	BRGEN	Res.	BLANKING[2:0]				HYST[1:0]	
rs	r				rw	rw		rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL[1:0]		INMSEL[2:0]		PWRMODE[1:0]	Res.	EN		
rw							rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 **LOCK**: COMP1\_CSR register lock bit

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the comparator 1 control register, COMP1\_CSR[31:0].

0: COMP1\_CSR[31:0] for comparator 1 are read/write

1: COMP1\_CSR[31:0] for comparator 1 are read-only

Bit 30 **VALUE**: Comparator 1 output status bit

This bit is read-only. It reflects the current comparator 1 output taking into account POLARITY bit effect.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:25 **INMESEL[1:0]**: Comparator 1 input minus extended selection bits.

These bits are set and cleared by software (only if LOCK is not set). They select which extended GPIO input is connected to the input minus of comparator if INMSEL = 111.

00: PC4 (available only on STM32WB55xx)

01: PA0

10: PA4

11: PA5

Bit 24 Reserved, must be kept at reset value.

Bit 23 **SCALEN**: Voltage scaler enable bit

This bit is set and cleared by software. This bit enable the outputs of the  $V_{REFINT}$  divider available on the minus input of the Comparator 1.

0: Bandgap scaler disable (if SCALEN bit of COMP2\_CSR register is also reset)

1: Bandgap scaler enable

Bit 22 **BRGEN**: Scaler bridge enable

This bit is set and cleared by software (only if LOCK not set). This bit enable the bridge of the scaler.

0: Scaler resistor bridge disable (if BRGEN bit of COMP2\_CSR register is also reset)

1: Scaler resistor bridge enable

If SCALEN is set and BRGEN is reset, BG voltage reference is available but not 1/4 BGAP, 1/2 BGAP, 3/4 BGAP. BGAP value is sent instead of 1/4 BGAP, 1/2 BGAP, 3/4 BGAP.

If SCALEN and BRGEN are set, 1/4 BGAP 1/2 BGAP 3/4 BGAP and BGAP voltage references are available.

## Bit 21 Reserved, must be kept at reset value.

Bits 20:18 **BLANKING[2:0]**: Comparator 1 blanking source selection bits

These bits select which timer output controls the comparator 1 output blanking.

000: No blanking

001: TIM1 OC5 selected as blanking source

010: TIM2 OC3 selected as blanking source

All other values: reserved

Bits 17:16 **HYST[1:0]**: Comparator 1 hysteresis selection bits

These bits are set and cleared by software (only if LOCK not set). They select the hysteresis voltage of the comparator 1.

00: No hysteresis

01: Low hysteresis

10: Medium hysteresis

11: High hysteresis

Bit 15 **POLARITY**: Comparator 1 polarity selection bit

This bit is set and cleared by software (only if LOCK not set). It inverts Comparator 1 polarity.

0: Comparator 1 output value not inverted

1: Comparator 1 output value inverted

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

Bits 8:7 **INPSEL[1:0]**: Comparator1 input plus selection bit

This bit is set and cleared by software (only if LOCK not set).

00: External I/O - PC5 (available only on STM32WB55xx)

01: PB2

10: PA2

11: Reserved

Bits 6:4 **INMSEL[2:0]**: Comparator 1 input minus selection bits

These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of comparator 1.

000 = 1/4  $V_{REFINT}$

001 = 1/2  $V_{REFINT}$

010 = 3/4  $V_{REFINT}$

011 =  $V_{REFINT}$

100 = Reserved

101 = Reserved

110 = PA9

111 = GPIOx selected by INMESEL bits

Bits 3:2 **PWRMODE[1:0]**: Power Mode of the comparator 1

These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the Comparator 1.

- 00: High speed
- 01 or 10: Medium speed
- 11: Ultra low power

Bit 1 Reserved, must be kept at reset value.

Bit 0 **EN**: Comparator 1 enable bit

This bit is set and cleared by software (only if LOCK not set). It switches on Comparator1.

- 0: Comparator 1 switched OFF
- 1: Comparator 1 switched ON

## 18.6.2 Comparator 2 control and status register (COMP2\_CSR)

The COMP2\_CSR is the Comparator 2 control/status register. It contains all the bits /flags related to comparator 2.

Address offset: 0x04

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	INMESEL[1:0]	Res.	SCAL EN	BRG EN	Res.	BLANKING[2:0]				HYST[1:0]	
rs	r				rw	rw		rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY	Res.	Res.	Res.	Res.	Res.	WIN MODE	INPSEL[1:0]		INMSEL[2:0]		PWRMODE[1:0]	Res.	EN		
rw						rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 **LOCK**: CSR register lock bit

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the comparator 2 control register, COMP2\_CSR[31:0].

- 0: COMP2\_CSR[31:0] for comparator 2 are read/write
- 1: COMP2\_CSR[31:0] for comparator 2 are read-only

Bit 30 **VALUE**: Comparator 2 output status bit

This bit is read-only. It reflects the current comparator 2 output taking into account POLARITY bit effect.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:25 **INMESEL[1:0]**: comparator 2 input minus extended selection bits.

These bits are set and cleared by software (only if LOCK is not set). They select which extended GPIO input is connected to the input minus of comparator if INMSEL = 111.

- 00: PB7
- 01: PA2
- 10: PA4
- 11: PA5

Bit 24 Reserved, must be kept at reset value.

Bit 23 **SCALEN**: Voltage scaler enable bit

This bit is set and cleared by software. This bit enable the outputs of the  $V_{REFINT}$  divider available on the minus input of the Comparator 2.

- 0: Bandgap scaler disable (if SCALEN bit of COMP1\_CSR register is also reset)
- 1: Bandgap scaler enable

Bit 22 **BRGEN**: Scaler bridge enable

This bit is set and cleared by software (only if LOCK not set). This bit enable the bridge of the scaler.

- 0: Scaler resistor bridge disable (if BRGEN bit of COMP1\_CSR register is also reset)
- 1: Scaler resistor bridge enable

If SCALEN is set and BRGEN is reset, BG voltage reference is available but not 1/4 BGAP, 1/2 BGAP, 3/4 BGAP. BGAP value is sent instead of 1/4 BGAP, 1/2 BGAP, 3/4 BGAP.

If SCALEN and BRGEN are set, 1/4 BGAP 1/2 BGAP 3/4 BGAP and BGAP voltage references are available.

Bit 21 Reserved, must be kept at reset value.

Bits 20:18 **BLANKING[2:0]**: Comparator 2 blanking source selection bits

These bits select which timer output controls the comparator 2 output blanking.

- 000: No blanking
- 001: TIM1 OC5 selected as blanking source
- 010: TIM2 OC3 selected as blanking source
- All other values: reserved

Bits 17:16 **HYST[1:0]**: Comparator 2 hysteresis selection bits

These bits are set and cleared by software (only if LOCK not set). Select the hysteresis voltage of the comparator 2.

- 00: No hysteresis
- 01: Low hysteresis
- 10: Medium hysteresis
- 11: High hysteresis

Bit 15 **POLARITY**: Comparator 2 polarity selection bit

This bit is set and cleared by software (only if LOCK not set). It inverts Comparator 2 polarity.

- 0: Comparator 2 output value not inverted
- 1: Comparator 2 output value inverted

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

Bit 9 **WINMODE**: Windows mode selection bit

This bit is set and cleared by software (only if LOCK not set). This bit selects the window mode of the comparators. If set, both positive inputs of comparators are connected together.

- 0: Input plus of Comparator 2 is not connected to Comparator 1
- 1: Input plus of Comparator 2 is connected with input plus of Comparator 1

Bits 8:7 **INPSEL[1:0]**: Comparator 2 input plus selection bit

This bit is set and cleared by software (only if LOCK not set).

- 00: PB4
- 01: PB6
- 10: PA3
- 11: Reserved

Bits 6:4 **INMSEL[2:0]**: Comparator 2 input minus selection bits

These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of comparator 2.

000 = 1/4  $V_{REFINT}$

001 = 1/2  $V_{REFINT}$

010 = 3/4  $V_{REFINT}$

011 =  $V_{REFINT}$

100 = Reserved

101 = Reserved

110 = PB3

111 = GPIOx selected by INMESEL bits

Bits 3:2 **PWRMODE[1:0]**: Power Mode of the comparator 2

These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the Comparator 2.

00: High speed

01 or 10: Medium speed

11: Ultra low power

Bit 1 Reserved, must be kept at reset value.

Bit 0 **EN**: Comparator 2 enable bit

This bit is set and cleared by software (only if LOCK not set). It switches oncomparator2.

0: Comparator 2 switched OFF

1: Comparator 2 switched ON

### 18.6.3 COMP register map

The following table summarizes the comparator registers.

**Table 101. COMP register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	<b>COMP1_CSR</b>	LOCK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		Reset value	Res.																														
0x04	<b>COMP2_CSR</b>	LOCK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 19 Liquid crystal display controller (LCD)

This section applies only to STM32WB55xx devices.

### 19.1 LCD introduction

The LCD controller is a digital controller/driver for monochrome passive liquid crystal display (LCD) with up to 8 common terminals and up to 44 segment terminals to drive 176 (44x4) or 320 (40x8) LCD picture elements (pixels). The exact number of terminals depends on the device pinout as described in the datasheet.

The LCD is made up of several segments (pixels or complete symbols) that can be turned visible or invisible. Each segment consists of a layer of liquid crystal molecules aligned between two electrodes. When a voltage greater than a threshold voltage is applied across the liquid crystal, the segment becomes visible. The segment voltage must be alternated to avoid an electrophoresis effect in the liquid crystal (which degrades the display).

The waveform across a segment must then be generated so as to avoid having a direct current (DC).

#### Glossary

**Bias:** number of voltage levels used when driving an LCD. It is defined as  $1 / (\text{number of voltage levels used to drive an LCD display} - 1)$ .

**Boost circuit:** Contrast controller circuit

**Common:** electrical connection terminal connected to several segments (44 segments)

**Duty ratio:** number defined as  $1 / (\text{number of common terminals on a given LCD display})$

**Frame:** one period of the waveform written to a segment

**Frame rate:** number of frames per second (number of times the LCD segments are energized per second)

**LCD** (liquid crystal display): passive display panel with terminals leading directly to a segment

**Segment:** smallest viewing element (a single bar or dot that is used to help create a character on an LCD display)

### 19.2 LCD main features

- Highly flexible frame rate control
- Static, 1/2, 1/3, 1/4, and 1/8 duty supported
- Static, 1/2, 1/3, and 1/4 bias supported
- Double buffered memory allowing data in LCD\_RAM registers to be updated at any time by the application firmware without affecting the integrity of the data displayed
  - LCD data RAM of up to 16 x 32-bit registers which contain pixel information (active/inactive)
- Software selectable LCD output voltage (contrast) from  $V_{LCDmin}$  to  $V_{LCDmax}$
- No need for external analog components:

- step-up converter embedded to generate an internal  $V_{LCD}$  voltage higher than  $V_{DD}$
- software selection between external and internal  $V_{LCD}$  voltage source. In case of an external source, the internal boost circuit is disabled to reduce power consumption.
- resistive network embedded to generate intermediate  $V_{LCD}$  voltages
- structure of the resistive network configurable by software to adapt the power consumption to match the capacitive charge required by the LCD panel
- Integrated voltage output buffers for higher LCD driving capability.
- Contrast that can be adjusted using two different methods:
  - When using the internal step-up converter, the software can adjust  $V_{LCD}$  between  $V_{LCDmin}$  and  $V_{LCDmax}$ .
  - Programmable dead time (up to eight phase periods) between frames
- Full support of low-power modes: the LCD controller can be displayed in Sleep, Low-power run, Low-power sleep, and Stop modes or can be fully disabled to reduce power consumption.
- Built in phase inversion for reduced power consumption and EMI (electromagnetic interference)
- Start-of-frame interrupt to synchronize the software when updating the LCD data RAM.
- Blink capability:
  - Up to 1, 2, 3, 4, 8, or all pixels which can be programmed to blink at a configurable frequency
  - Software adjustable blink frequency to achieve around 0.5 Hz, 1 Hz, 2 Hz or 4 Hz
- Used LCD segment and common pins must be configured as GPIO alternate functions, and unused segment and common pins can be used for GPIO or for another peripheral alternate function.

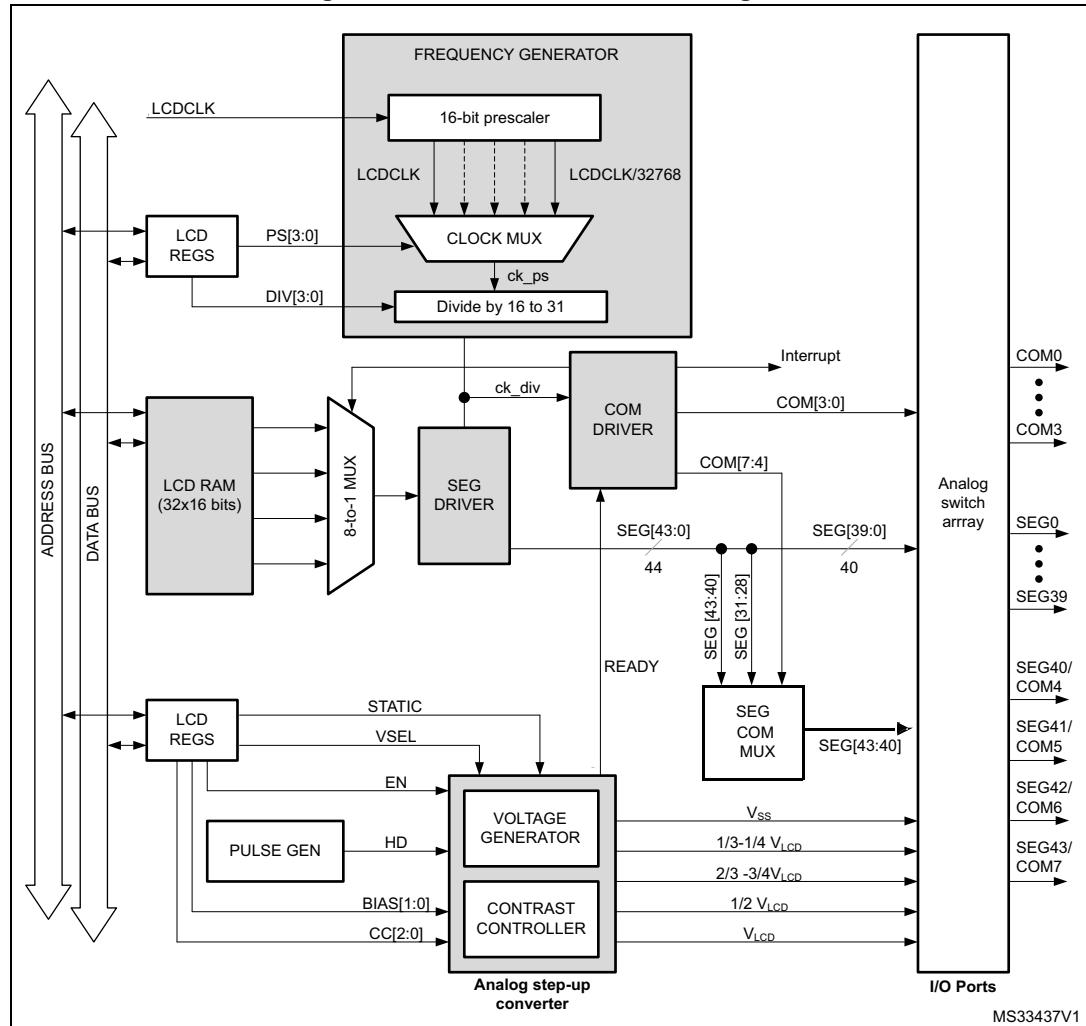
*Note:* When the LCD relies on the internal step-up converter, the  $V_{LCD}$  pin must be connected to  $V_{SS}$  with a capacitor. Its typical value is 1  $\mu$ F (see  $C_{EXT}$  value in the product datasheets for further information).

## 19.3 LCD functional description

### 19.3.1 General description

The LCD controller has five main blocks (see [Figure 96](#)):

**Figure 96. LCD controller block diagram**



**Note:** LCDCLK is the same as RTCCLK. Refer to the RTC/LCD clock description in the RCC section of the product reference manual.

The frequency generator allows the user to achieve various LCD frame rates starting from an LCD input clock frequency (LCDCLK) which can vary from 32 kHz up to 1 MHz.

3 different clock sources can be used to provide the LCD clock (LCDCLK/RTCCLK):

- 32 kHz low-speed external RC (LSE)
- 32 kHz low-speed internal RC (LSI)
- High-speed external (HSE) divided by 32

### 19.3.2 Frequency generator

This clock source must be stable in order to obtain accurate LCD timing, and hence to minimize DC voltage offset across LCD segments. The input clock LCDCLK can be divided by any value from 1 to  $2^{15} \times 31$  (see [Section 19.6.2](#)). The frequency generator consists of a prescaler (16-bit ripple counter) and a 16 to 31 clock divider. The PS[3:0] bits, in the LCD\_FCR register, select LCDCLK divided by  $2^{\text{PS}[3:0]}$ . If a finer resolution rate is required, the DIV[3:0] bits, in the LCD\_FCR register, can be used to divide the clock further by 16 to 31. In this way, the user can roughly scale the frequency, and then fine-tune it by linearly scaling the clock with the counter. The output of the frequency generator block is  $f_{\text{ck\_div}}$  which constitutes the time base for the entire LCD controller. The ck\_div frequency is equivalent to the LCD phase frequency, rather than the frame frequency (they are equal only in case of static duty). The frame frequency ( $f_{\text{frame}}$ ) is obtained from  $f_{\text{ck\_div}}$  by dividing it by the number of active common terminals (or by multiplying it for the duty). Thus the relation between the input clock frequency ( $f_{\text{LCDCLK}}$ ) of the frequency generator and its output clock frequency  $f_{\text{ck\_div}}$  is:

$$f_{\text{ckdiv}} = \frac{f_{\text{LCDCLK}}}{2^{\text{PS}} \times (16 + \text{DIV})}$$

$$f_{\text{frame}} = f_{\text{ckdiv}} \times \text{duty}$$

This makes the frequency generator very flexible. An example of frame rate calculation is shown in the table below.

**Table 102. Example of frame rate calculation**

LCDCLK	PS[3:0]	DIV[3:0]	Ratio	Duty	$f_{\text{frame}}$
32.768 kHz	3	1	136	1/8	30.12 Hz
32.768 kHz	4	1	272	1/4	30.12 Hz
32.768 kHz	4	6	352	1/3	31.03 Hz
32.768 kHz	5	1	544	1/2	30.12 Hz
32.768 kHz	6	1	1088	static	30.12 Hz
32.768 kHz	1	4	40	1/8	102.40 Hz
32.768 kHz	2	4	80	1/4	102.40 Hz
32.768 kHz	2	11	108	1/3	101.14 Hz
32.768 kHz	3	4	160	1/2	102.40 Hz
32.768 kHz	4	4	320	static	102.40 Hz
1.00 MHz	6	3	1216	1/8	102.80 Hz
1.00 MHz	7	3	2432	1/4	102.80 Hz
1.00 MHz	7	10	3328	1/3	100.16 Hz
1.00 MHz	8	3	4864	1/2	102.80 Hz
1.00 MHz	9	3	9728	static	102.80 Hz

The frame frequency must be selected to be within a range of around ~30 Hz to ~100 Hz. It is a compromise between power consumption and the acceptable refresh rate. In addition, a dedicated blink prescaler selects the blink frequency. This frequency is defined as:

$$f_{BLINK} = f_{ck\_div}/2^{(BLINKF + 3)},$$

with  $\text{BLINKF}[2:0] = 0, 1, 2, \dots, 7$

The blink frequency achieved is in the range of 0.5 Hz, 1 Hz, 2 Hz, or 4 Hz.

### 19.3.3 Common driver

Common signals are generated by the common driver block (see [Figure 96](#)).

## COM signal bias

Each COM signal has identical waveforms, but different phases. It has its max amplitude  $V_{LCD}$  or  $V_{SS}$  only in the corresponding phase of a frame cycle, while during the other phases, the signal amplitude is:

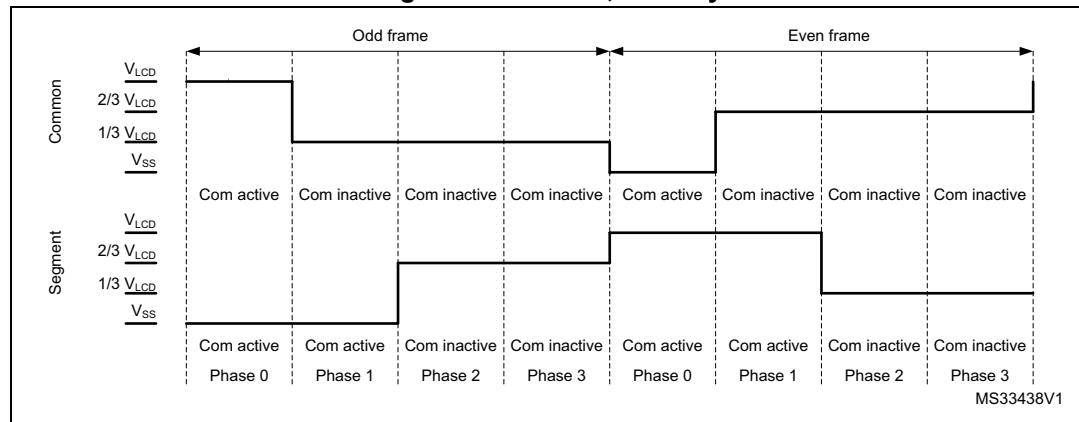
- $1/4 V_{LCD}$  or  $3/4 V_{LCD}$  in case of  $1/4$  bias
  - $1/3 V_{LCD}$  or  $2/3 V_{LCD}$  in case of  $1/3$  bias
  - $1/2 V_{LCD}$  in case of  $1/2$  bias

The selection between 1/2, 1/3 and 1/4 bias mode can be done through BIAS bits in the LCD\_CR register.

A pixel is activated when both of its corresponding common and segment lines are active during the same phase: when the voltage difference between common and segment is maximum during this phase. Common signals are phase inverted in order to reduce EMI.

As shown in the figure below, with phase inversion, there is a mean voltage of  $1/2 V_{LCD}$  at the end of every odd cycle.

**Figure 97. 1/3 bias, 1/4 duty**



In case of 1/2 bias (BIAS = 01), the VLCD pin generates an intermediate voltage equal to 1/2  $V_{LCD}$  on node b for odd and even frames (see [Figure 100](#)).

## COM signal duty

Depending on the DUTY[2:0] bits in the LCD\_CR register, the COM signals are generated with static duty (see [Figure 99](#)), 1/2 duty (see [Figure 100](#)), 1/3 duty (see [Figure 101](#)), 1/4 duty (see [Figure 102](#)) or 1/8 duty (see [Figure 103](#)).

COM[n] n[0 to 7] is active during phase n in the odd frame, so the COM pin is driven to  $V_{LCD}$ .

During phase n of the even frame the COM pin is driven to  $V_{SS}$ .

In the case of 1/3 or 1/4) bias, COM[n] is inactive during phases other than n so the COM pin is driven to 1/3 (1/4)  $V_{LCD}$  during odd frames and to 2/3 (3/4)  $V_{LCD}$  during even frames.

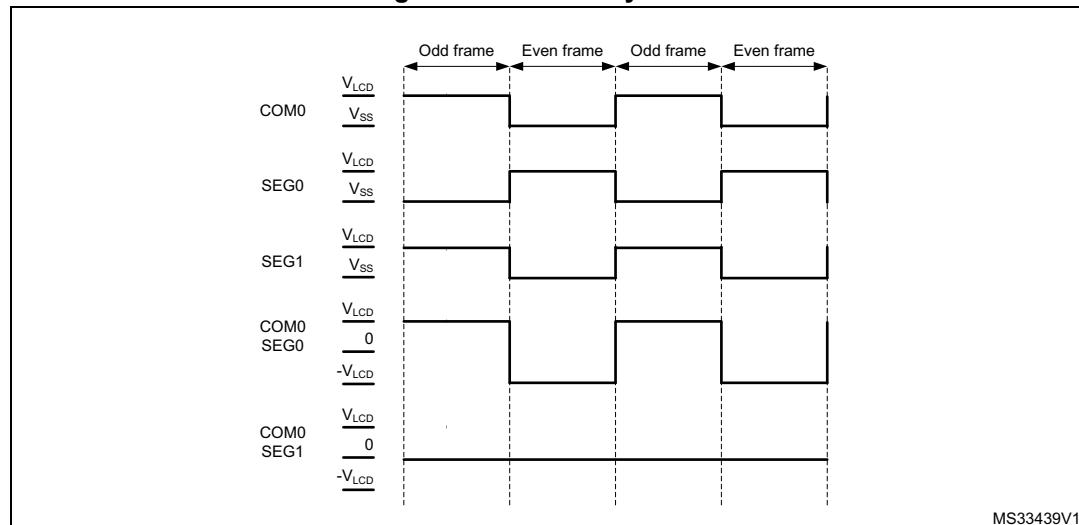
In the case of 1/2 bias, if COM[n] is inactive during phases other than n, the COM pin is always driven (odd and even frame) to 1/2  $V_{LCD}$ .

When static duty is selected, the segment lines are not multiplexed, which means that each segment output corresponds to one pixel. In this way only up to 44 pixels can be driven. COM[0] is always active while COM[7:1] are not used and are driven to  $V_{SS}$ .

When the LCDEN bit in the LCD\_CR register is reset, all common lines are pulled down to  $V_{SS}$  and the ENS flag in the LCD\_SR register becomes 0. Static duty means that COM[0] is always active and only two voltage levels are used for the segment and common lines:  $V_{LCD}$  and  $V_{SS}$ . A pixel is active if the corresponding SEG line has a voltage opposite to that of the COM, and inactive when the voltages are equal. In this way the LCD has maximum contrast (see [Figure 98](#), [Figure 99](#)).

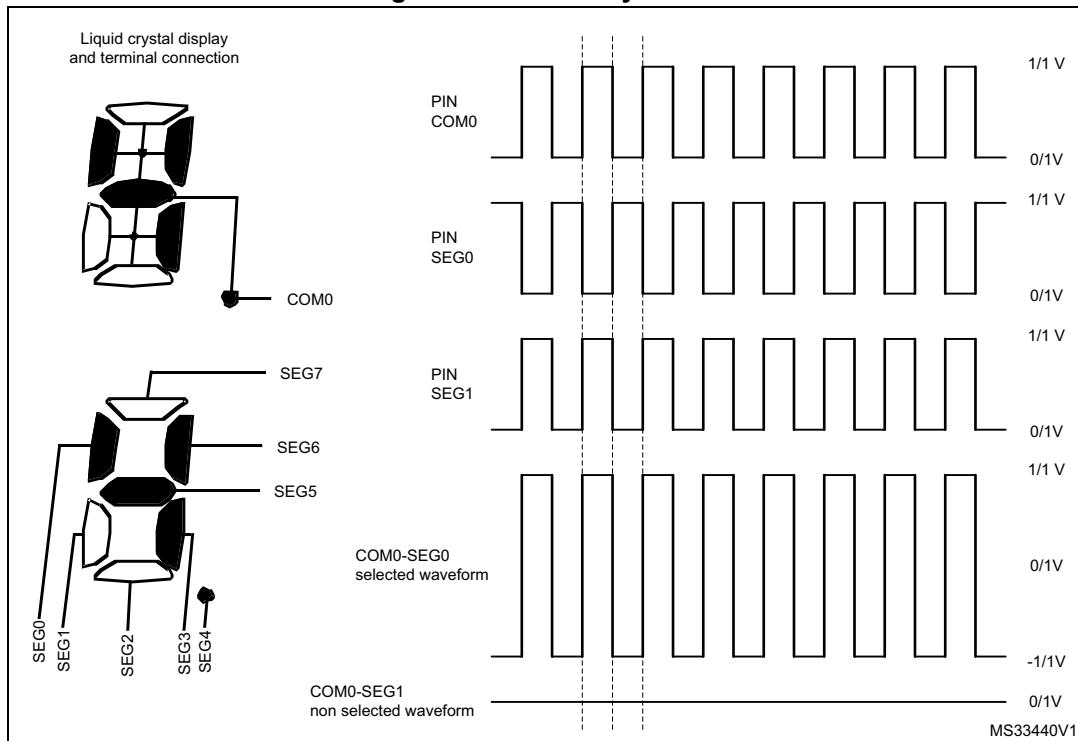
In the figure below, pixel 0 is active while pixel 1 is inactive.

**Figure 98. Static duty case 1**



In each frame there is only one phase, this is why  $f_{frame}$  is equal to  $f_{LCD}$ . If 1/4 duty is selected there are four phases in a frame in which COM[0] is active during phase 0, COM[1] is active during phase 1, COM[2] is active during phase 2, and COM[3] is active during phase 3.

Figure 99. Static duty case 2

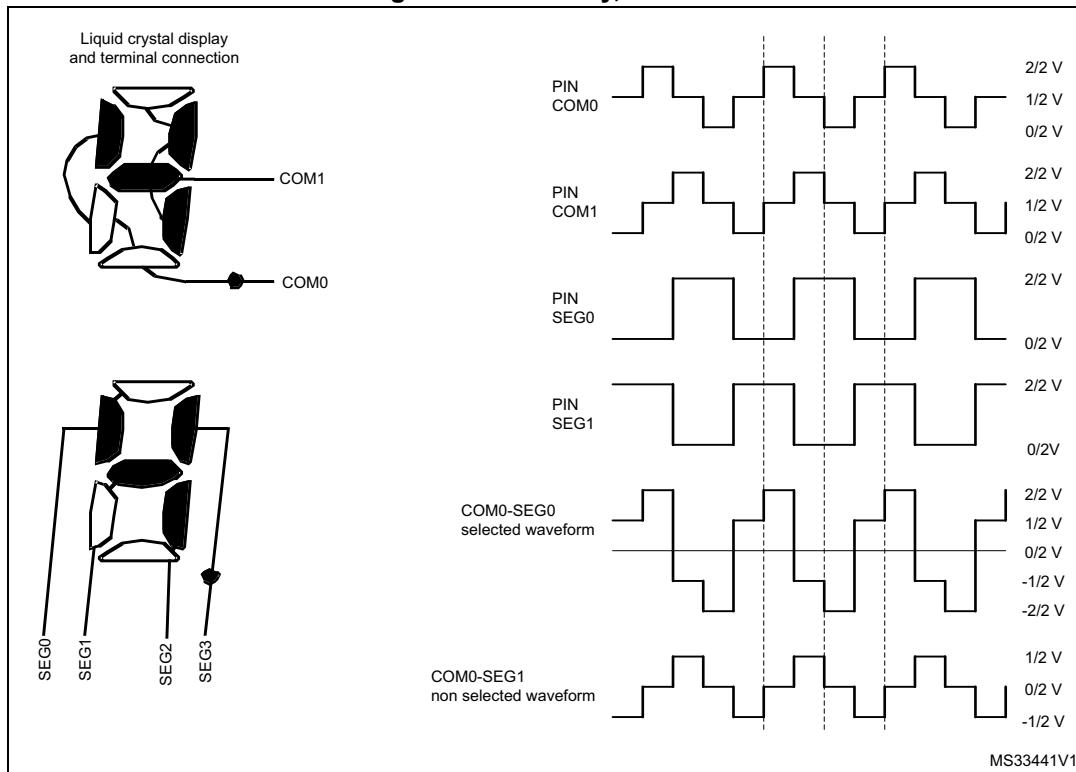


In this mode, the segment terminals are multiplexed and each of them control four pixels. A pixel is activated only when both of its corresponding SEG and COM lines are active in the same phase. In case of 1/4 duty, to deactivate pixel 0 connected to COM[0] the SEG[0] needs to be inactive during the phase 0 when COM[0] is active. To activate pixel 0 connected to COM[1], the SEG[0] needs to be active during phase 1 when COM[1] is active (see [Figure 102](#)). To activate pixels from 0 to 43 connected to COM[0], SEG[0:43] need to be active during phase 0 when COM[0] is active. These considerations can be extended to the other pixels.

### Eight-to-one mux

When COM[0] is active the common driver block, also drives the eight-to-one mux shown in [Figure 96](#) in order to select the content of first two RAM register locations. When COM[7] is active, the output of the eight-to-one mux is the content of the last two RAM locations.

Figure 100. 1/2 duty, 1/2 bias



### 19.3.4 Segment driver

The segment driver block controls the SEG lines according to the pixel data coming from the eight-to-one mux driven in each phase by the common driver block.

#### In the case of 1/4 or 1/8 duty

When COM[0] is active, the pixel information (active/inactive) related to the pixel connected to COM[0] (content of the first two LCD\_RAM locations) goes through the eight-to-one mux.

The SEG[n] pin n [0 to 43] is driven to  $V_{SS}$  (indicating pixel n is active when COM[0] is active) in phase 0 of the odd frame.

The SEG[n] pin is driven to  $V_{LCD}$  in phase 0 of the even frame. If pixel n is inactive then the SEG[n] pin is driven to  $2/3$  ( $2/4$ )  $V_{LCD}$  in the odd frame or  $1/3$  ( $2/4$ )  $V_{LCD}$  in the even frame (current inversion in  $V_{LCD}$  pad) (see [Figure 97](#)).

In case of 1/2 bias, if the pixel is inactive the SEG[n] pin is driven to  $V_{LCD}$  in the odd and to  $V_{SS}$  in the even frame.

When the LCD controller is disabled (LCDEN cleared in LCD\_CR), then the SEG lines are pulled down to  $V_{SS}$ .

Figure 101. 1/3 duty, 1/3 bias

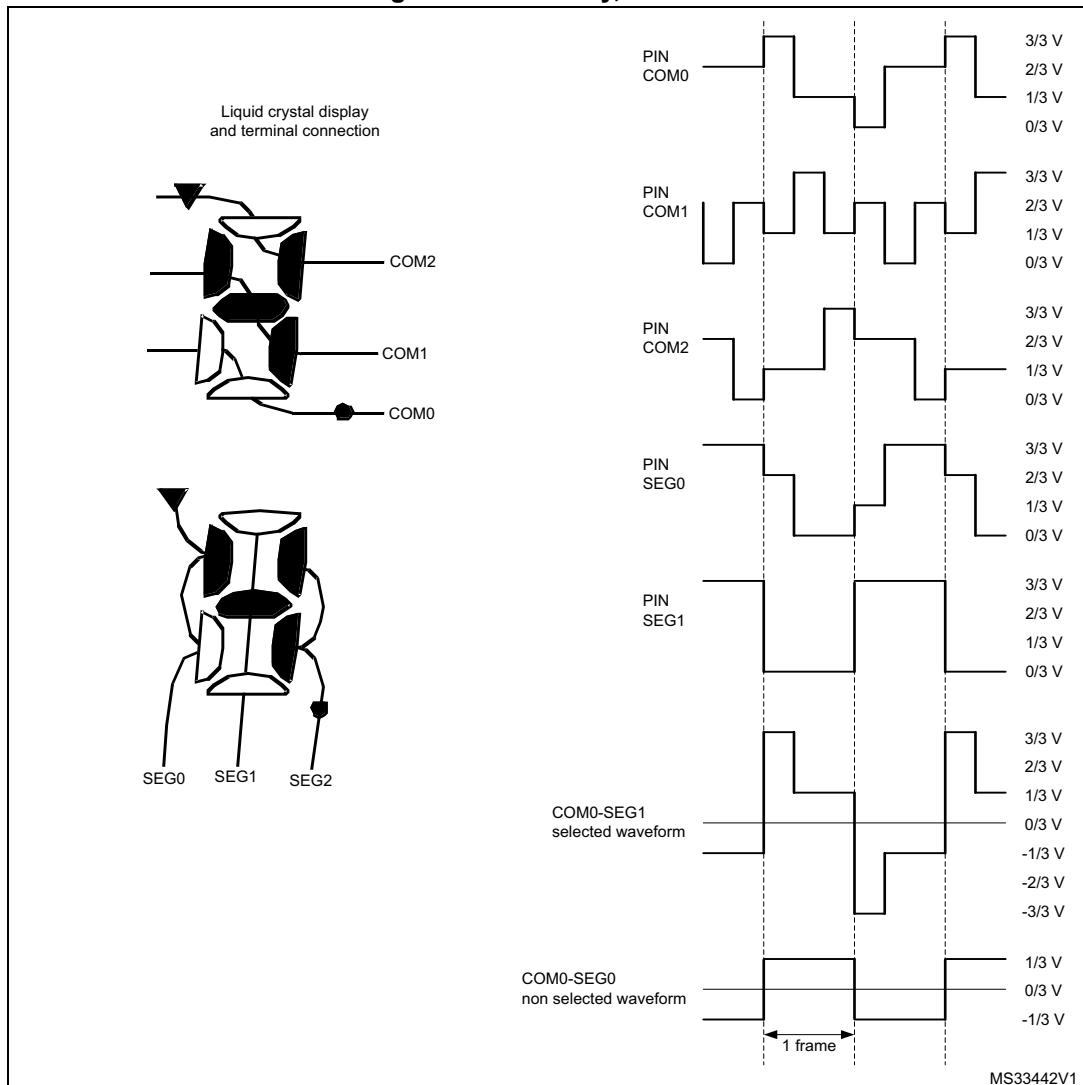


Figure 102. 1/4 duty, 1/3 bias

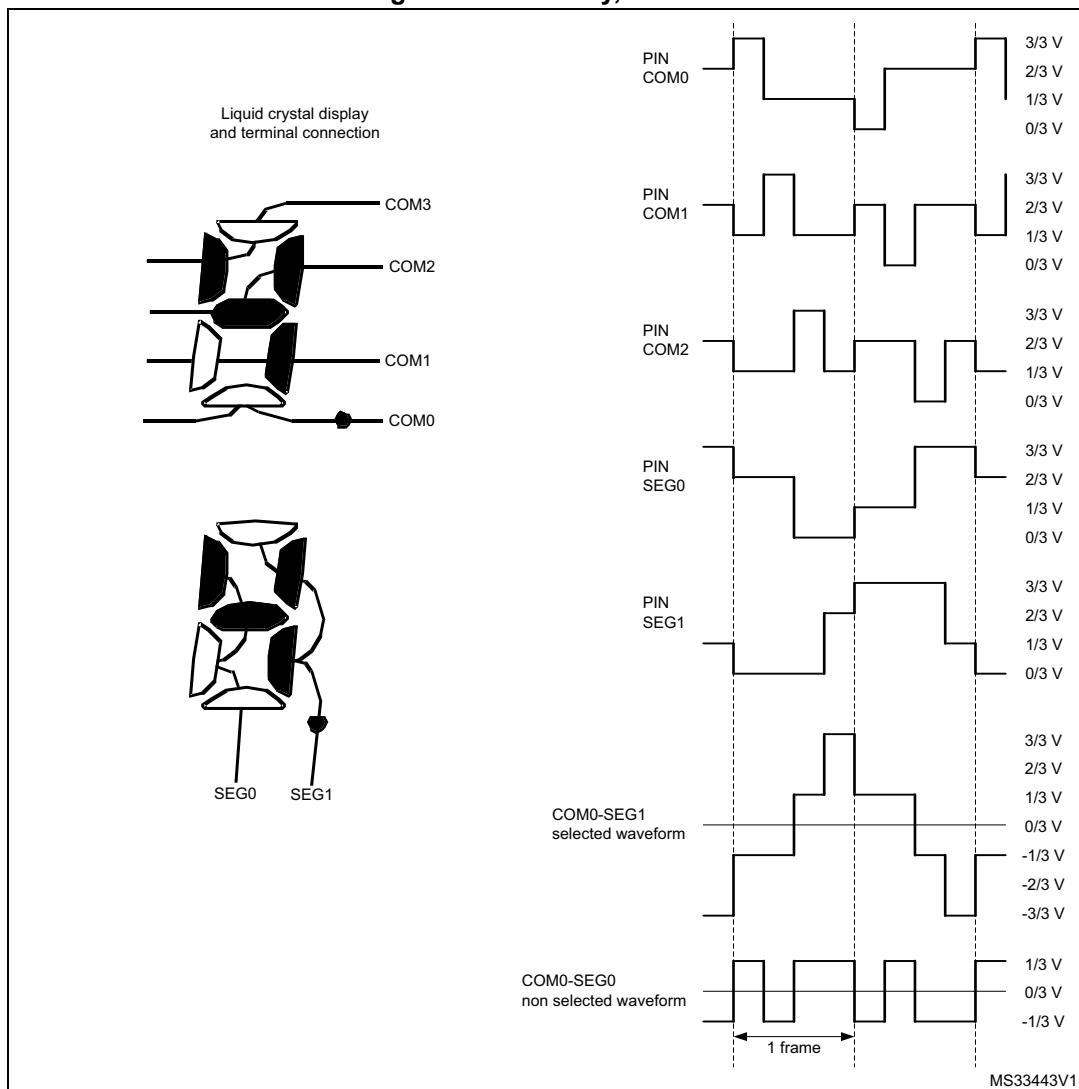
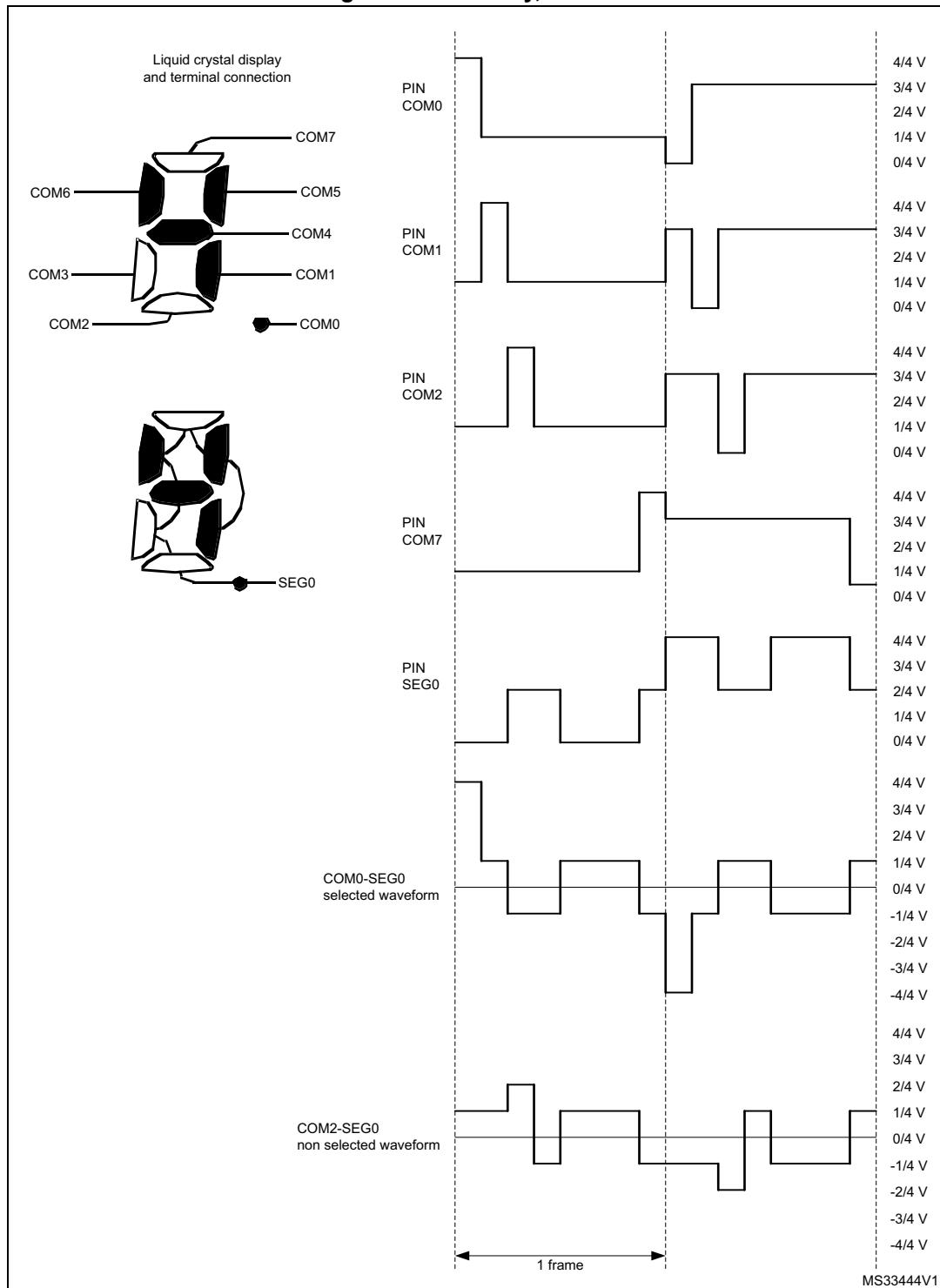


Figure 103. 1/8 duty, 1/4 bias



## Blink

The segment driver also implements a programmable blink feature to allow some pixels to continuously switch on at a specific frequency. The blink mode can be configured by the BLINK[1:0] bits in the LCD\_FCR register, making possible to blink up to 1, 2, 4, 8, or all pixels (see [Section 19.6.2: LCD frame control register \(LCD\\_FCR\)](#)). The blink frequency can be selected from eight different values using BLINKF[2:0] in LCD\_FCR.

The table below gives examples of different blink frequencies (as a function of ck\_div frequency).

**Table 103. Blink frequency**

BLINKF[2:0] bits			ck_div frequency (with LCDCLK frequency of 32.768 kHz)			
			32 Hz	64 Hz	128 Hz	256 Hz
0	0	0	4.0 Hz	N/A	N/A	N/A
0	0	1	2.0 Hz	4.0 Hz	N/A	N/A
0	1	0	1.0 Hz	2.0 Hz	4.0 Hz	N/A
0	1	1	0.5 Hz	1.0 Hz	2.0 Hz	4.0 Hz
1	0	0	0.25 Hz	0.5 Hz	1.0 Hz	2.0 Hz
1	0	1	N/A	0.25 Hz	0.5 Hz	1.0 Hz
1	1	0	N/A	N/A	0.25 Hz	0.5 Hz
1	1	1	N/A	N/A	N/A	0.25 Hz

### 19.3.5 Voltage generator and contrast control

#### LCD supply source

The LCD power supply source comes from either the internal step-up converter or from an external voltage applied on the VLCD pin. The internal or external voltage source can be selected using VSEL bit in LCD\_CR. In case of external source selected, the internal boost circuit (step-up converter) is disabled to reduce power consumption.

When the step-up converter is selected as  $V_{LCD}$  source, the  $V_{LCD}$  value can be chosen among a wide set of values from  $V_{LCDmin}$  to  $V_{LCDmax}$  by means of CC[2:0] (Contrast Control) bits inside LCD\_FCR (see [Section 19.6.2: LCD frame control register \(LCD\\_FCR\)](#)) register. New values of  $V_{LCD}$  takes effect every beginning of a new frame.

When external power source is selected as  $V_{LCD}$  source, the  $V_{LCD}$  voltage must be chosen in the range of  $V_{LCDmin}$  to  $V_{LCDmax}$  (see datasheets). The contrast can then be controlled by programming a dead time between frames (see [Deadtime on page 538](#)).

A specific software sequence must be performed to configure the LCD depending on the LCD power supply source to be used. Here we consider the LCD controller is disabled prior to the configuration sequence.

In case the internal step-up converter is used (capacitor  $C_{EXT}$  on VLCD pin is required):

- Configure the VLCD pin as alternate function LCD in the GPIO\_AFR register.
- Wait for the external capacitor  $C_{EXT}$  to be charged ( $C_{EXT}$  connected to the VLCD pin, approximately 2 ms for  $C_{EXT} = 1 \mu F$ )

- Set voltage source to internal source by resetting VSEL bit in the LCD\_CR register
- Enable the LCD controller by setting LCDEN bit in the LCD\_CR register

In case of LCD external power source is used:

- Set voltage source to external source by setting VSEL bit in the LCD\_CR register
- Configure the VLCD pin as alternate function LCD in the GPIO\_AFR register
- Enable the LCD controller by setting LCDEN bit in the LCD\_CR register

### LCD intermediate voltages

The LCD intermediate voltage levels are generated through an internal resistor divider network as shown in [Figure 104](#).

The LCD voltage generator issues intermediate voltage levels between  $V_{SS}$  and  $V_{LCD}$ :

- 1/3  $V_{LCD}$  and 2/3  $V_{LCD}$  in case of 1/3 bias
- 1/4  $V_{LCD}$ , 2/4  $V_{LCD}$  and 3/4  $V_{LCD}$  in case of 1/4 bias
- only 1/2  $V_{LCD}$  in case of 1/2 bias.

### LCD drive selection

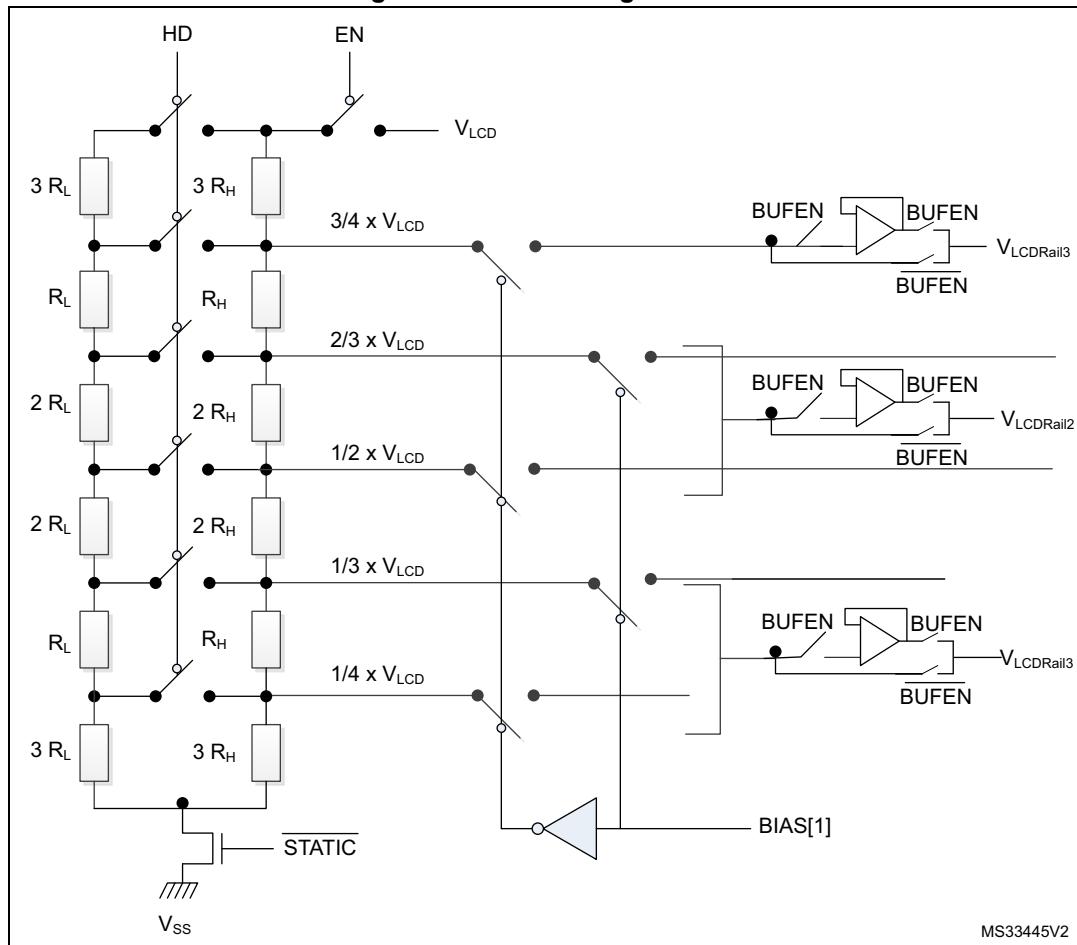
Two resistive networks, one with low value resistors ( $R_L$ ) and one with high value resistors ( $R_H$ ) are respectively used to increase the current during transitions and reduce power consumption in static state.

The EN switch follows the rules described below (see [Figure 104](#)):

- If LCDEN bit in the LCD\_CR register is set, the EN switch is closed.
- When clearing the LCDEN bit in the LCD\_CR register, the EN switch is open at the end of the even frame in order to avoid a medium voltage level different from 0 considering the entire frame odd plus even.

The PON[2:0] (Pulse ON duration) bits in the LCD\_FCR register configure the time during which  $R_L$  is enabled through the HD (high drive) switch when the levels of the common and segment lines change (see [Figure 104](#)). A short drive time leads to lower power consumption, but displays with high internal resistance may need a longer drive time to achieve satisfactory contrast.

Figure 104. LCD voltage control



MS33445V2

1.  $R_{LN}$  and  $R_{HN}$  are the low value resistance network and the high value resistance network, respectively.

The  $R_{LN}$  divider can be always switched on using the HD bit in the LCD\_FCR configuration register (see [Section 19.6.2](#)).

The HD switch follows the rules described below:

- If the HD bit and the PON[2:0] bits in the LCD\_FCR register are reset, then HD switch is open.
- If the HD bit in the LCD\_FCR register is reset and the PON[2:0] bits in the LCD\_FCR are different from 00 then, the HD switch is closed during the number of pulses defined in the PON[2:0] bits.
- If HD bit in the LCD\_FCR register is 1 then HD switch is always closed.

### Buffered mode

When voltage output buffers are enabled by setting BUFEN bit in the LCD\_CR register, LCD driving capability is improved as buffers prevent the LCD capacitive loads from loading the resistor bridge unacceptably and interfering with its voltage generation. As a result we obtain more stable intermediate voltage levels thus improving RMS voltage applied to the LCD pixels.

In buffer mode, intermediate voltages are generated by the high value resistor bridge  $R_{HN}$  to reduce power consumption, the low value resistor bridge  $R_{LN}$  is automatically disabled whatever the HD bit or PON bits configuration.

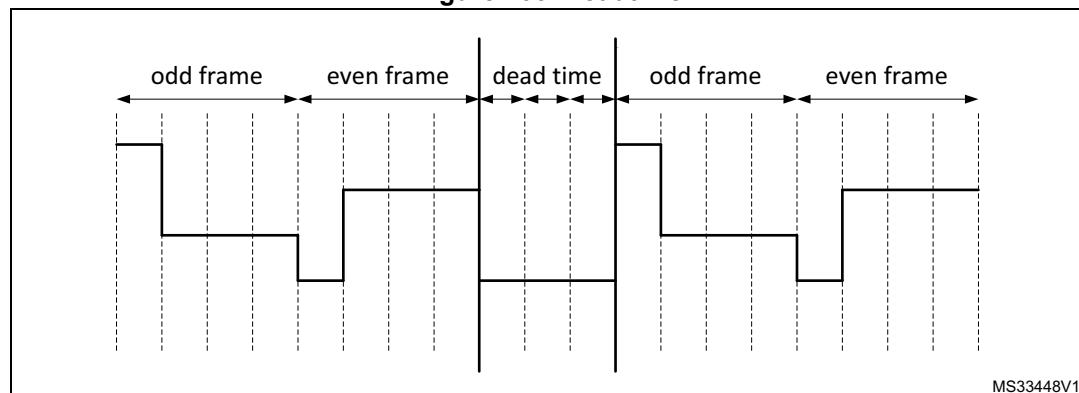
Buffers can be used independently of the  $V_{LCD}$  supply source (internal or external) but can only be enabled or disabled when LCD controller is not activated.

After the LCDEN bit is activated, the RDY bit is set in the LCD\_SR register to indicate that voltage levels are stable and the LCD controller can start to work.

### Deadtime

In addition to using the CC[2:0] bits, the contrast can be controlled by programming a dead time between each frame. During the dead time the COM and SEG values are put to  $V_{SS}$ . The DEAD[2:0] bits in the LCD\_FCR register can be used to program a time of up to eight phase periods. This dead time reduces the contrast without modifying the frame rate.

Figure 105. Deadtime



### 19.3.6 Double buffer memory

Using its double buffer memory the LCD controller ensures the coherency of the displayed information without having to use interrupts to control LCD\_RAM modification.

The application software can access the first buffer level (LCD\_RAM) through the APB interface. Once it has modified the LCD\_RAM, it sets the UDR flag in the LCD\_SR register. This UDR flag (update display request) requests the updated information to be moved into the second buffer level (LCD\_DISPLAY).

This operation is done synchronously with the frame (at the beginning of the next frame), until the update is completed, the LCD\_RAM is write protected and the UDR flag stays high. Once the update is completed another flag (UDD - Update Display Done) is set and generates an interrupt if the UDDIE bit in the LCD\_FCR register is set.

The time it takes to update LCD\_DISPLAY is, in the worst case, one odd and one even frame.

The update does not occur (UDR = 1 and UDD = 0) until the display is enabled (LCDEN = 1)

### 19.3.7 COM and SEG multiplexing

#### Output pins versus duty modes

The output pins consists of:

- SEG[43:0]
- COM[3:0]

Depending on the duty configuration, the COM and SEG output pins may have different functions:

- In static, 1/2, 1/3 and 1/4 duty modes there are up to 44 SEG pins and respectively 1, 2, 3 and 4 COM pins
- In 1/8 duty mode (DUTY[2:0] = 100), COM[7:4] outputs are available on the SEG[43:40] pins, reducing to the number of available segments to 40.

#### Remapping capability for small packages

Additionally, it is possible to remap 4 segments by setting the MUX\_SEG bit in the LCD\_CR register. This is particularly useful when using smaller device types with fewer external pins. When MUX\_SEG is set, output pins SEG[43:40] have the same function as SEG[31:28].

This feature is available only if the mode 1/8 duty is not selected.

Check the availability of this feature referring to the pinout section of the product datasheet.

For the considered package, check the availability of the SEG/COM multiplexing pin as follow:

LCD SEG[n-1]/LCD COM7/LCD SEG[31]

LCD SEG[n-2]/LCD COM6/LCD SEG[30]

LCD SEG[n-3]/LCD COM5/LCD SEG[29]

LCD SEG[n-4]/LCD COM4/LCD SEG[28]

with n = number of segment for the considered package.

#### Summary of COM and SEG functions versus duty and remap

All the possible ways of multiplexing the COM and SEG functions are described in [Table 104](#). [Figure 106](#) gives examples showing the signal connections to the external pins.

**Table 104. Remapping capability**

DUTY	MUX_SEG	WLCSP100	VFQFPN68	UFQFPN48	Output pin	Function
1/8	n.a.	40x8	-	-	COM[3:0]	COM[3:0]
					SEG[43:40]/SEG[31:28]/COM[7:4]	COM[7:4]
					SEG[39:0]	SEG[39:0]

Table 104. Remapping capability (continued)

DUTY	MUX_SEG	WLCSP100	VFQFPN68	UFQFPN48	Output pin	Function
1/4	0	44x4	-	-	COM[3:0]	COM[3:0]
					SEG[43:40]/SEG[31:28]/COM[7:4]	SEG[43:40]
					SEG[39:0]	SEG[39:0]
	1	40x4	-	-	COM[3:0]	COM[3:0]
					SEG[43:40]/SEG[31:28]/COM[7:4]	SEG[31:28]
					SEG[39:32]	SEG[39:32]
					SEG[31:28]	not used
					SEG[27:0]	SEG[27:0]
	0	-	28x4	-	COM[3:0]	COM[3:0]
					SEG[42:40]/SEG[30:28]/COM[6:4]	SEG[42:40]
					SEG[24:0]	SEG[24:0]
	1	-	28x4	-	COM[3:0]	COM[3:0]
					SEG[42:40]/SEG[30:28]/COM[6:4]	SEG[30:28]
					SEG[24:0]	SEG[24:0]
	n.a.	-	-	13x4	COM[3:0]	COM[3:0]
					SEG[21]	SEG[21]
					SEG[17:16]	SEG[17:16]
					SEG[9:0]	SEG[9:0]

Table 104. Remapping capability (continued)

DUTY	MUX_SEG	WLCSP100	VFQFPN68	UFQFPN48	Output pin	Function
1/3	0	44x3	-	-	COM[3]	not used
					COM[2:0]	COM[2:0]
					SEG[43:40]/SEG[31:28]/COM[7:4]	SEG[43:40]
					SEG[39:0]	SEG[39:0]
	1	40x3	-	-	COM[3]	not used
					COM[2:0]	COM[2:0]
					SEG[43:40]/SEG[31:28]/COM[7:4]	SEG[31:28]
					SEG[39:32]	SEG[39:32]
					SEG[31:28]	not used
					SEG[27:0]	SEG[27:0]
	0	-	28x3	-	COM[3]	not used
					COM[2:0]	COM[2:0]
					SEG[42:40]/SEG[30:28]/COM[6:4]	SEG[42:40]
					SEG[24:0]	SEG[24:0]
	1	-	28x3	-	COM[3]	not used
					COM[2:0]	COM[2:0]
					SEG[42:40]/SEG[30:28]/COM[6:4]	SEG[30:28]
					SEG[24:0]	SEG[24:0]
	n.a.	-	-	13x3	COM[3]	not used
					COM[2:0]	COM[2:0]
					SEG[21]	SEG[21]
					SEG[17:16]	SEG[17:16]
					SEG[9:0]	SEG[9:0]
1/2	0	44x2	-	-	COM[3:2]	not used
					COM[1:0]	COM[1:0]
					SEG[43:40]/SEG[31:28]/COM[7:4]	SEG[43:40]
					SEG[39:0]	SEG[39:0]
	1	40x2	-	-	COM[3:2]	not used
					COM[1:0]	COM[1:0]
					SEG[43:40]/SEG[31:28]/COM[7:4]	SEG[31:28]
					SEG[39:32]	SEG[39:32]
					SEG[31:28]	not used
					SEG[27:0]	SEG[27:0]

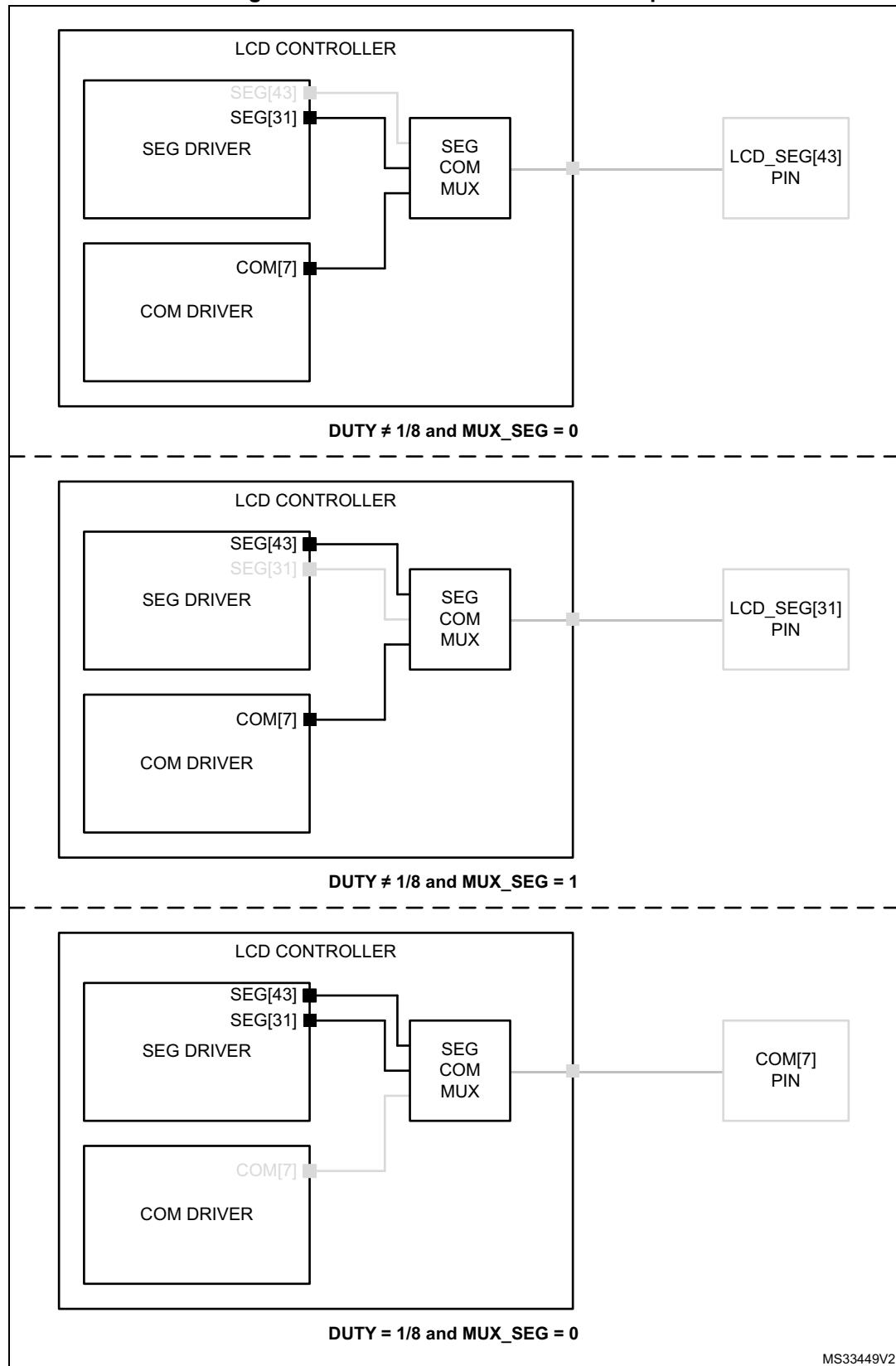
Table 104. Remapping capability (continued)

DUTY	MUX_SEG	WLCSP100	VFQFPN68	UFQFPN48	Output pin	Function
1/2	0	-	28x2	-	COM[3:2]	not used
					COM[1:0]	COM[1:0]
					SEG[42:40]/SEG[30:28]/COM[6:4]	SEG[42:40]
					SEG[24:0]	SEG[24:0]
	1	-	28x2	-	COM[3:2]	not used
					COM[1:0]	COM[1:0]
					SEG[42:40]/SEG[30:28]/COM[6:4]	SEG[30:28]
					SEG[24:0]	SEG[24:0]
	n.a.	-	-	13x2	COM[3:2]	not used
					COM[1:0]	COM[1:0]
					SEG[21]	SEG[21]
					SEG[17:16]	SEG[17:16]
					SEG[9:0]	SEG[9:0]
STATIC	0	44x1	-	-	COM[3:1]	not used
					COM[0]	COM[0]
					SEG[43:40]/SEG[31:28]/COM[7:4]	SEG[43:40]
					SEG[39:0]	SEG[39:0]
	1	40x1	-	-	COM[3:1]	not used
					COM[0]	COM[0]
					SEG[43:40]/SEG[31:28]/COM[7:4]	SEG[31:28]
					SEG[39:32]	SEG[39:32]
					SEG[31:28]	not used
					SEG[27:0]	SEG[27:0]
	0	-	28x1	-	COM[3:1]	not used
					COM[0]	COM[0]
					SEG[42:40]/SEG[30:28]/COM[6:4]	SEG[42:40]
					SEG[24:0]	SEG[24:0]
	1	-	28x1	-	COM[3:1]	not used
					COM[0]	COM[0]
					SEG[42:40]/SEG[30:28]/COM[6:4]	SEG[30:28]
					SEG[24:0]	SEG[24:0]

Table 104. Remapping capability (continued)

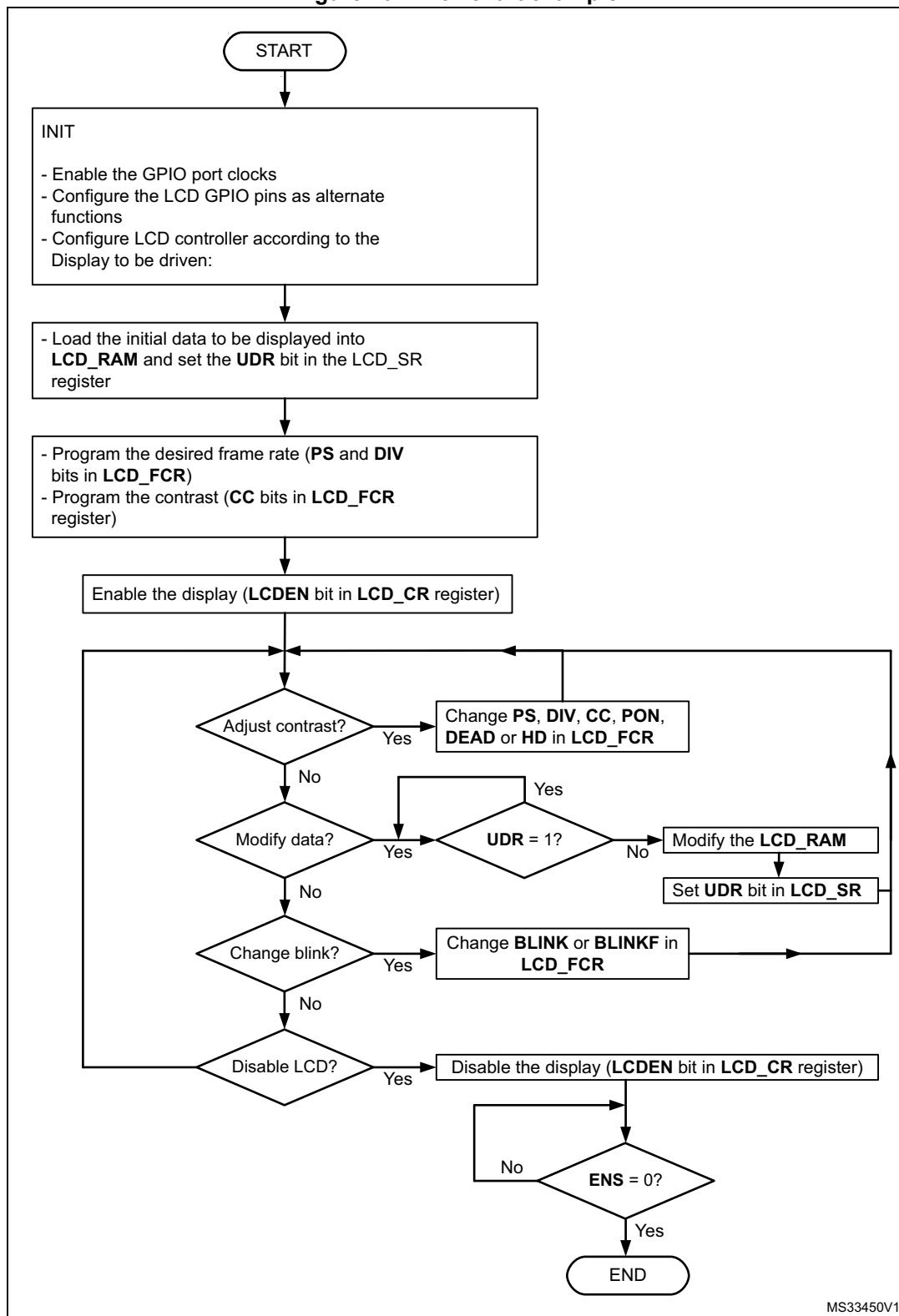
DUTY	MUX_SEG	WLCSP100	VFQFPN68	UFQFPN48	Output pin	Function
STATIC	n.a.	-	-	13x1	COM[3:1]	not used
					COM[0]	COM[0]
					SEG[21]	SEG[21]
					SEG[17:16]	SEG[17:16]
					SEG[9:0]	SEG[9:0]

Figure 106. SEG/COM mux feature example



### 19.3.8 Flowchart

Figure 107. Flowchart example



MS33450V1

## 19.4 LCD low-power modes

the LCD controller can be displayed in Stop mode, or can be fully disabled to reduce power consumption.

## 19.5 LCD interrupts

The table below gives the list of LCD interrupt requests.

**Table 105. LCD interrupt requests**

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Start Of Frame (SOF)	SOF	Write SOFC =1	SOFIE
Update Display Done (UDD)	UDD	Write UDDC = 1	UDDIE

### Start of frame (SOF)

The LCD start of frame interrupt is executed if the SOFIE (start of frame interrupt enable) bit is set (see [Section 19.6.2](#)). SOF is cleared by writing the SOFC bit to 1 in the LCD\_CLR register when executing the corresponding interrupt handling vector.

### Update display done (UDD)

The LCD update display interrupt is executed if the UDDIE (update display done interrupt enable) bit is set (see [Section 19.6.2](#)). UDD is cleared by writing the UDDC bit to 1 in the LCD\_CLR register when executing the corresponding interrupt handling vector.

Depending on the product implementation, all these interrupts events can either share the same interrupt vector (LCD global interrupt), or be grouped into 2 interrupt vectors (LCD SOF interrupt and LCD UDD interrupt). Refer to the [Table 61: CPU1 vector table](#) for details.

To enable the LCD interrupts, the following sequence is required:

1. Configure and enable the LCD IRQ channel in the NVIC
2. Configure the LCD to generate interrupts

## 19.6 LCD registers

The peripheral registers have to be accessed by words (32-bit).

### 19.6.1 LCD control register (LCD\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

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

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

Bit 8 **BUFEN**: Voltage output buffer enable

This bit is used to enable/disable the voltage output buffer for higher driving capability.

- 0: Output buffer disabled
- 1: Output buffer enabled

Bit 7 **MUX\_SEG**: Mux segment enable

This bit is used to enable SEG pin remapping. Four SEG pins can be multiplexed with SEG[31:28]. See [Section 19.3.7](#).

- 0: SEG pin multiplexing disabled
- 1: SEG[31:28] are multiplexed with SEG[43:40]

Bits 6:5 **BIAS[1:0]**: Bias selector

These bits determine the bias used. Value 11 is forbidden.

- 00: Bias 1/4
- 01: Bias 1/2
- 10: Bias 1/3
- 11: Reserved

Bits 4:2 **DUTY[2:0]**: Duty selection

These bits determine the duty cycle. Values 101, 110 and 111 are forbidden.

- 000: Static duty
- 001: 1/2 duty
- 010: 1/3 duty
- 011: 1/4 duty
- 100: 1/8 duty
- Others: Reserved

Bit 1 **VSEL**: Voltage source selection

The VSEL bit determines the voltage source for the LCD.

- 0: Internal source (voltage step-up converter)
- 1: External source (VLCD pin)

Bit 0 **LCDEN**: LCD controller enable

This bit is set by software to enable the LCD Controller/Driver. It is cleared by software to turn off the LCD at the beginning of the next frame. When the LCD is disabled all COM and SEG pins are driven to  $V_{SS}$ .

- 0: LCD controller disabled
- 1: LCD controller enabled

**Note:** *The VSEL, MUX\_SEG,BIAS, DUTY and BUFEN bits are write-protected when the LCD is enabled (ENS bit in LCD\_SR to 1).*

## 19.6.2 LCD frame control register (LCD\_FCR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS[3:0]				DIV[3:0]				BLINK[1:0]
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLINKF[2:0]				CC[2:0]				DEAD[2:0]				PON[2:0]		UDDIE	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
												SOFIE	HD		

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:22 **PS[3:0]**: PS 16-bit prescaler

These bits are written by software to define the division factor of the PS 16-bit prescaler.

$ck_{ps} = LCDCLK/(2)$ . See [Section 19.3.2](#).

- 0000:  $ck_{ps} = LCDCLK$
- 0001:  $ck_{ps} = LCDCLK/2$
- 0002:  $ck_{ps} = LCDCLK/4$
- ...

1111:  $ck_{ps} = LCDCLK/32768$

Bits 21:18 **DIV[3:0]**: DIV clock divider

These bits are written by software to define the division factor of the DIV divider. See [Section 19.3.2](#).

- 0000:  $ck_{div} = ck_{ps}/16$
- 0001:  $ck_{div} = ck_{ps}/17$
- 0002:  $ck_{div} = ck_{ps}/18$
- ...

1111:  $ck_{div} = ck_{ps}/31$

Bits 17:16 **BLINK[1:0]**: Blink mode selection

- 00: Blink disabled
- 01: Blink enabled on SEG[0], COM[0] (1 pixel)
- 10: Blink enabled on SEG[0], all COMs (up to 8 pixels depending on the programmed duty)
- 11: Blink enabled on all SEGs and all COMs (all pixels)

Bits 15:13 **BLINKF[2:0]**: Blink frequency selection

000:  $f_{LCD}/8$   
001:  $f_{LCD}/16$   
010:  $f_{LCD}/32$   
011:  $f_{LCD}/64$   
100:  $f_{LCD}/128$   
101:  $f_{LCD}/256$   
110:  $f_{LCD}/512$   
111:  $f_{LCD}/1024$

Bits 12:10 **CC[2:0]**: Contrast control

These bits specify one of the  $V_{LCD}$  maximum voltages (independent of  $V_{DD}$ ). It ranges from 2.60 V to 3.51V.

000:  $V_{LCD0}$   
001:  $V_{LCD1}$   
010:  $V_{LCD2}$   
011:  $V_{LCD3}$   
100:  $V_{LCD4}$   
101:  $V_{LCD5}$   
110:  $V_{LCD6}$   
111:  $V_{LCD7}$

Refer to the product datasheet for the  $V_{LCDx}$  values.

Bits 9:7 **DEAD[2:0]**: Dead time duration

These bits are written by software to configure the length of the dead time between frames. During the dead time the COM and SEG voltage levels are held at 0 V to reduce the contrast without modifying the frame rate.

000: No dead time  
001: 1 phase period dead time  
010: 2 phase period dead time  
.....  
111: 7 phase period dead time

Bits 6:4 **PON[2:0]**: Pulse ON duration

These bits are written by software to define the pulse duration in terms of ck\_ps pulses. A short pulse leads to lower power consumption, but displays with high internal resistance may need a longer pulse to achieve satisfactory contrast.

Note that the pulse is never longer than one half prescaled LCD clock period.

000: 0

001: 1/ck\_ps

010: 2/ck\_ps

011: 3/ck\_ps

100: 4/ck\_ps

101: 5/ck\_ps

110: 6/ck\_ps

111: 7/ck\_ps

PON duration example with LCDCLK = 32.768 kHz and PS=0x03:

000: 0  $\mu$ s

001: 244  $\mu$ s

010: 488  $\mu$ s

011: 782  $\mu$ s

100: 976  $\mu$ s

101: 1.22 ms

110: 1.46 ms

111: 1.71 ms

Bit 3 **UDDIE**: Update display done interrupt enable

This bit is set and cleared by software.

0: LCD Update display done interrupt disabled

1: LCD Update display done interrupt enabled

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

Bit 1 **SOFIE**: Start of frame interrupt enable

This bit is set and cleared by software.

0: LCD start-of-frame interrupt disabled

1: LCD start-of-frame interrupt enabled

Bit 0 **HD**: High drive enable

This bit is written by software to enable a low resistance divider. Displays with high internal resistance may need a longer drive time to achieve satisfactory contrast. This bit is useful in this case if some additional power consumption can be tolerated.

0: Permanent high drive disabled

1: Permanent high drive enabled. When HD=1, then the PON bits have to be programmed to 001.

**Note:** *The data in this register can be updated any time, however the new values are applied only at the beginning of the next frame (except for UDDIE, SOFIE that affect the device behavior immediately).*

*The new value of CC[2:0] bits is also applied immediately but its effect on device is delayed at the beginning of next frame by the voltage generator.*

*Reading this register obtains the last value written in the register and not the configuration used to display the current frame.*

**Note:** *When BUFEN bit is set in the LCD\_CR register, low resistor divider network is automatically disabled whatever the HD or PON[2:0] bits configuration.*

### 19.6.3 LCD status register (LCD\_SR)

Address offset: 0x08

Reset value: 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FCRSF	RDY	UDD	UDR	SOF	ENS									
										r	r	r	rs	r	r

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

**Bit 5 FCRSF:** LCD frame control register synchronization flag

This bit is set by hardware each time the LCD\_FCR register is updated in the LCDCLK domain. It is cleared by hardware when writing to the LCD\_FCR register.

- 0: LCD frame control register not yet synchronized
- 1: LCD frame control register synchronized

**Bit 4 RDY:** Ready flag

This bit is set and cleared by hardware. It indicates the status of the step-up converter.

- 0: Not ready
- 1: Step-up converter is enabled and ready to provide the correct voltage.

**Bit 3 UDD:** Update display done

This bit is set by hardware. It is cleared by writing 1 to the UDDC bit in the LCD\_CLR register. The bit set has priority over the clear.

- 0: No event
- 1: Update Display Request done. A UDD interrupt is generated if the UDDIE bit in the LCD\_FCR register is set.

*Note: If the device is in Stop mode (PCLK not provided), UDD does not generate an interrupt even if UDDIE = 1. If the display is not enabled the UDD interrupt never occurs.*

**Bit 2 UDR:** Update display request

Each time software modifies the LCD\_RAM it must set the UDR bit to transfer the updated data to the second level buffer. The UDR bit stays set until the end of the update and during this time the LCD\_RAM is write protected.

- 0: No effect
- 1: Update display request

*Note: When the display is disabled, the update is performed for all LCD\_DISPLAY locations.*

*When the display is enabled, the update is performed only for locations for which commons are active (depending on DUTY). For example if DUTY = 1/2, only the LCD\_DISPLAY of COM0 and COM1 is updated.*

*Note: Writing 0 on this bit or writing 1 when it is already 1 has no effect. This bit can be cleared by hardware only. It can be cleared only when LCDEN = 1*

**Bit 1 SOF:** Start-of-frame flag

This bit is set by hardware at the beginning of a new frame, at the same time as the display data is updated. It is cleared by writing a 1 to the SOFC bit in the LCD\_CLR register. The bit clear has priority over the set.

- 0: No event
- 1: Start-of-frame event occurred. An LCD start-of-frame interrupt is generated if SOFIE is set.

Bit 0 **ENS**: LCD enabled status

This bit is set and cleared by hardware. It indicates the LCD controller status.

0: LCD controller disabled

1: LCD controller enabled

*Note: The ENS bit is set immediately when the LCDEN bit in the LCD\_CR goes from 0 to 1.*

*On deactivation it reflects the real status of LCD so it becomes 0 at the end of the last displayed frame.*

#### 19.6.4 LCD clear register (LCD\_CLR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bit 3 **UDDC**: Update display done clear

This bit is written by software to clear the UDD flag in the LCD\_SR register.

0: No effect

1: Clear UDD flag

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SOFC**: Start of frame flag clear

This bit is written by software to clear the SOF flag in the LCD\_SR register.

0: No effect

1: Clear SOF flag

Bit 0 Reserved, must be kept at reset value.

#### 19.6.5 LCD display memory (LCD\_RAM)

Address offset: 0x14 to 0x50

Reset value: 0x0000 0000

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

Bits 31:0 **SEGMENT\_DATA[31:0]**: Each bit corresponds to one pixel of the LCD display.

0: Pixel inactive

1: Pixel active

## 19.6.6 LCD register map

Table 106. LCD register map and reset values

		Offset	Register name		
0x00				LCD_CR	
0x00	0x00	0x00	Reset value	Res.	Res.
0x04	0x04	0x04	LCD_FCR	Res.	Res.
0x08	0x08	0x08	Reset value	Res.	Res.
0x0C	0x0C	0x0C	LCD_SR	Res.	Res.
0x14	0x14	0x14	Reset value	Res.	Res.
0x18	0x18	0x18	LCD_CLR	Res.	Res.
0x20	0x20	0x20	Reset value	Res.	Res.
0x24	0x24	0x24	LCD_RAM (COM0)	Res.	Res.
0x28	0x28	0x28	LCD_RAM (COM1)	Res.	Res.
0x2C	0x2C	0x2C	LCD_RAM (COM2)	Res.	Res.
0x30	0x30	0x30	LCD_RAM (COM3)	Res.	Res.
			S31	Res.	Res.
			S30	Res.	Res.
			S29	Res.	Res.
			S28	Res.	Res.
			S27	Res.	Res.
			S26	Res.	Res.
			S25	Res.	Res.
			S24	Res.	Res.
			S23	Res.	Res.
			S22	Res.	Res.
			S21	Res.	Res.
			S20	Res.	Res.
			S19	Res.	Res.
			S18	Res.	Res.
			S17	Res.	Res.
			S16	Res.	Res.
			S15	Res.	Res.
			S14	Res.	Res.
			S13	Res.	Res.
			S12	Res.	Res.
			S11	0	0
			S10	0	0
			S09	0	0
			S08	0	0
			S07	0	0
			S06	0	0
			S05	0	0
			S04	0	0
			S03	0	0
			S02	0	0
			S01	0	0
			S00	0	0
			S37	0	0
			S36	0	0
			S35	0	0
			S34	0	0
			S33	0	0
			S32	0	0
			S31	0	0
			S30	0	0
			S29	0	0
			S28	0	0
			S27	0	0
			S26	0	0
			S25	0	0
			S24	0	0
			S23	0	0
			S22	0	0
			S21	0	0
			S20	0	0
			S19	0	0
			S18	0	0
			S17	0	0
			S16	0	0
			S15	0	0
			S14	0	0
			S13	0	0
			S12	0	0
			S11	0	0
			S10	0	0
			S09	0	0
			S08	0	0
			S07	0	0
			S06	0	0
			S05	0	0
			S04	0	0
			S03	0	0
			S02	0	0
			S01	0	0
			S00	0	0
			S37	0	0
			S36	0	0
			S35	0	0
			S34	0	0
			S33	0	0
			S32	0	0
			S31	0	0
			S30	0	0
			S29	0	0
			S28	0	0
			S27	0	0
			S26	0	0
			S25	0	0
			S24	0	0
			S23	0	0
			S22	0	0
			S21	0	0
			S20	0	0
			S19	0	0
			S18	0	0
			S17	0	0
			S16	0	0
			S15	0	0
			S14	0	0
			S13	0	0
			S12	0	0
			S11	0	0
			S10	0	0
			S09	0	0
			S08	0	0
			S07	0	0
			S06	0	0
			S05	0	0
			S04	0	0
			S03	0	0
			S02	0	0
			S01	0	0
			S00	0	0
			S37	0	0
			S36	0	0
			S35	0	0
			S34	0	0
			S33	0	0
			S32	0	0
			S31	0	0
			S30	0	0
			S29	0	0
			S28	0	0
			S27	0	0
			S26	0	0
			S25	0	0
			S24	0	0
			S23	0	0
			S22	0	0
			S21	0	0
			S20	0	0
			S19	0	0
			S18	0	0
			S17	0	0
			S16	0	0
			S15	0	0
			S14	0	0
			S13	0	0
			S12	0	0
			S11	0	0
			S10	0	0
			S09	0	0
			S08	0	0
			S07	0	0
			S06	0	0
			S05	0	0
			S04	0	0
			S03	0	0
			S02	0	0
			S01	0	0
			S00	0	0
			S37	0	0
			S36	0	0
			S35	0	0
			S34	0	0
			S33	0	0
			S32	0	0
			S31	0	0
			S30	0	0
			S29	0	0
			S28	0	0
			S27	0	0
			S26	0	0
			S25	0	0
			S24	0	0
			S23	0	0
			S22	0	0
			S21	0	0
			S20	0	0
			S19	0	0
			S18	0	0
			S17	0	0
			S16	0	0
			S15	0	0
			S14	0	0
			S13	0	0
			S12	0	0
			S11	0	0
			S10	0	0
			S09	0	0
			S08	0	0
			S07	0	0
			S06	0	0
			S05	0	0
			S04	0	0
			S03	0	0
			S02	0	0
			S01	0	0
			S00	0	0
			S37	0	0
			S36	0	0
			S35	0	0
			S34	0	0
			S33	0	0
			S32	0	0
			S31	0	0
			S30	0	0
			S29	0	0
			S28	0	0
			S27	0	0
			S26	0	0
			S25	0	0
			S24	0	0
			S23	0	0
			S22	0	0
			S21	0	0
			S20	0	0
			S19	0	0
			S18	0	0
			S17	0	0
			S16	0	0
			S15	0	0
			S14	0	0
			S13	0	0
			S12	0	0
			S11	0	0
			S10	0	0
			S09	0	0
			S08	0	0
			S07	0	0
			S06	0	0
			S05	0	0
			S04	0	0
			S03	0	0
			S02	0	0
			S01	0	0
			S00	0	0
			S37	0	0
			S36	0	0
			S35	0	0
			S34	0	0
			S33	0	0
			S32	0	0
			S31	0	0
			S30	0	0
			S29	0	0
			S28	0	0
			S27	0	0
			S26	0	0
			S25	0	0
			S24	0	0
			S23	0	0
			S22	0	0
			S21	0	0
			S20	0	0
			S19	0	0
			S18	0	0
			S17	0	0
			S16	0	0
			S15	0	0
			S14	0	0
			S13	0	0
			S12	0	0
			S11	0	0
			S10	0	0
			S09	0	0
			S08	0	0
			S07	0	0
			S06	0	0
			S05	0	0
			S04	0	0
			S03	0	0
			S02	0	0
			S01	0	0
			S00	0	0
			S37	0	0
			S36	0	0
			S35	0	0
			S34	0	0
			S33	0	0
			S32	0	0
			S31	0	0
			S30	0	0
			S29	0	0
			S28	0	0
			S27	0	0
			S26	0	0
			S25	0	0
			S24	0	0
			S23	0	0
			S22	0	0
			S21	0	0
			S20	0	0
			S19	0	0
			S18	0	0
			S17	0	0
			S16	0	0
			S15	0	0
			S14	0	0
			S13	0	0
			S12	0	0

Table 106. LCD register map and reset values (continued)

Offset	Register name	0x34	0x38	0x3C	0x40	0x44	0x48	0x4C	0x50
	LCD_RAM (COM4)								
	LCD_RAM (COM5)								
	LCD_RAM (COM6)								
	LCD_RAM (COM7)								
0	Res. 0 S31	0	0	0	0	0	0	0	0
0	Res. 0 S30	0	0	0	0	0	0	0	0
0	Res. 0 S29	0	0	0	0	0	0	0	0
0	Res. 0 S28	0	0	0	0	0	0	0	0
0	Res. 0 S27	0	0	0	0	0	0	0	0
0	Res. 0 S26	0	0	0	0	0	0	0	0
0	Res. 0 S25	0	0	0	0	0	0	0	0
0	Res. 0 S24	0	0	0	0	0	0	0	0
0	Res. 0 S23	0	0	0	0	0	0	0	0
0	Res. 0 S22	0	0	0	0	0	0	0	0
0	Res. 0 S21	0	0	0	0	0	0	0	0
0	Res. 0 S20	0	0	0	0	0	0	0	0
0	Res. 0 S19	0	0	0	0	0	0	0	0
0	Res. 0 S18	0	0	0	0	0	0	0	0
0	Res. 0 S17	0	0	0	0	0	0	0	0
0	Res. 0 S16	0	0	0	0	0	0	0	0
0	Res. 0 S15	0	0	0	0	0	0	0	0
0	Res. 0 S14	0	0	0	0	0	0	0	0
0	Res. 0 S13	0	0	0	0	0	0	0	0
0	Res. 0 S12	0	0	0	0	0	0	0	0
0	Res. 0 S11	0	0	0	0	0	0	0	0
0	Res. 0 S10	0	0	0	0	0	0	0	0
0	Res. 0 S09	0	0	0	0	0	0	0	0
0	Res. 0 S08	0	0	0	0	0	0	0	0
0	S39 0 S07	0	0	0	0	0	0	0	0
0	S38 0 S06	0	0	0	0	0	0	0	0
0	S37 0 S05	0	0	0	0	0	0	0	0
0	S36 0 S04	0	0	0	0	0	0	0	0
0	S35 0 S03	0	0	0	0	0	0	0	0
0	S34 0 S02	0	0	0	0	0	0	0	0
0	S33 0 S01	0	0	0	0	0	0	0	0
0	S32 0 S00	0	0	0	0	0	0	0	0

Refer to [Section 2.2](#) for the register boundary addresses table.

## 20 Touch sensing controller (TSC)

This section applies to STM32WB55xx devices only.

### 20.1 Introduction

The touch sensing controller provides a simple solution for adding capacitive sensing functionality to any application. Capacitive sensing technology is able to detect finger presence near an electrode that is protected from direct touch by a dielectric (for example glass, plastic). The capacitive variation introduced by the finger (or any conductive object) is measured using a proven implementation based on a surface charge transfer acquisition principle.

The touch sensing controller is fully supported by the STMTouch touch sensing firmware library, which is free to use and allows touch sensing functionality to be implemented reliably in the end application.

### 20.2 TSC main features

The touch sensing controller has the following main features:

- Proven and robust surface charge transfer acquisition principle
- Supports up to 28 capacitive sensing channels without shield
- Supports up to 6 capacitive sensing channels when shield is used in group 7 (recommended mode)
- Spread spectrum feature to improve system robustness in noisy environments
- Full hardware management of the charge transfer acquisition sequence
- Programmable charge transfer frequency
- Programmable sampling capacitor I/O pin
- Programmable channel I/O pin
- Programmable max count value to avoid long acquisition when a channel is faulty
- Dedicated end of acquisition and max count error flags with interrupt capability
- One sampling capacitor for up to 3 capacitive sensing channels to reduce the system components
- Compatible with proximity, touchkey, linear and rotary touch sensor implementation
- Designed to operate with STMTouch touch sensing firmware library

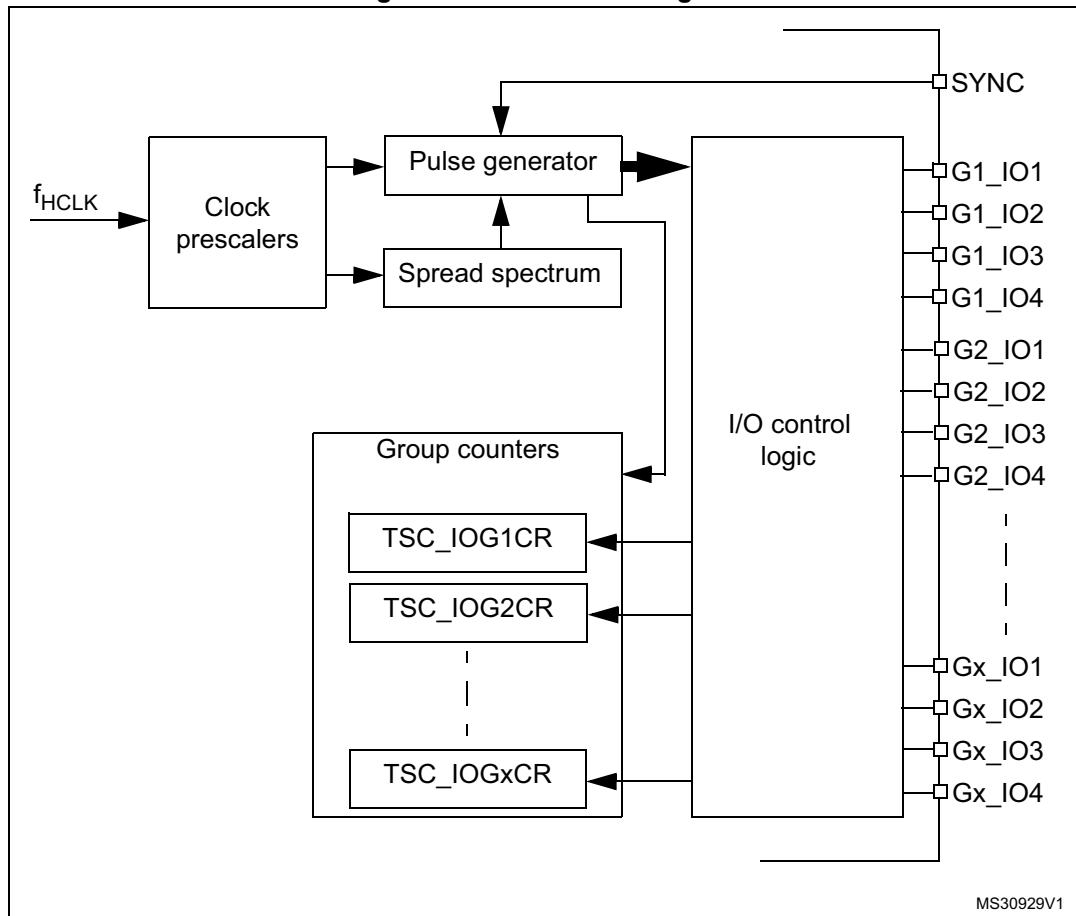
*Note:* The number of capacitive sensing channels is dependent on the size of the packages and subject to IO availability.

## 20.3 TSC functional description

### 20.3.1 TSC block diagram

The block diagram of the touch sensing controller is shown in [Figure 108](#).

**Figure 108. TSC block diagram**



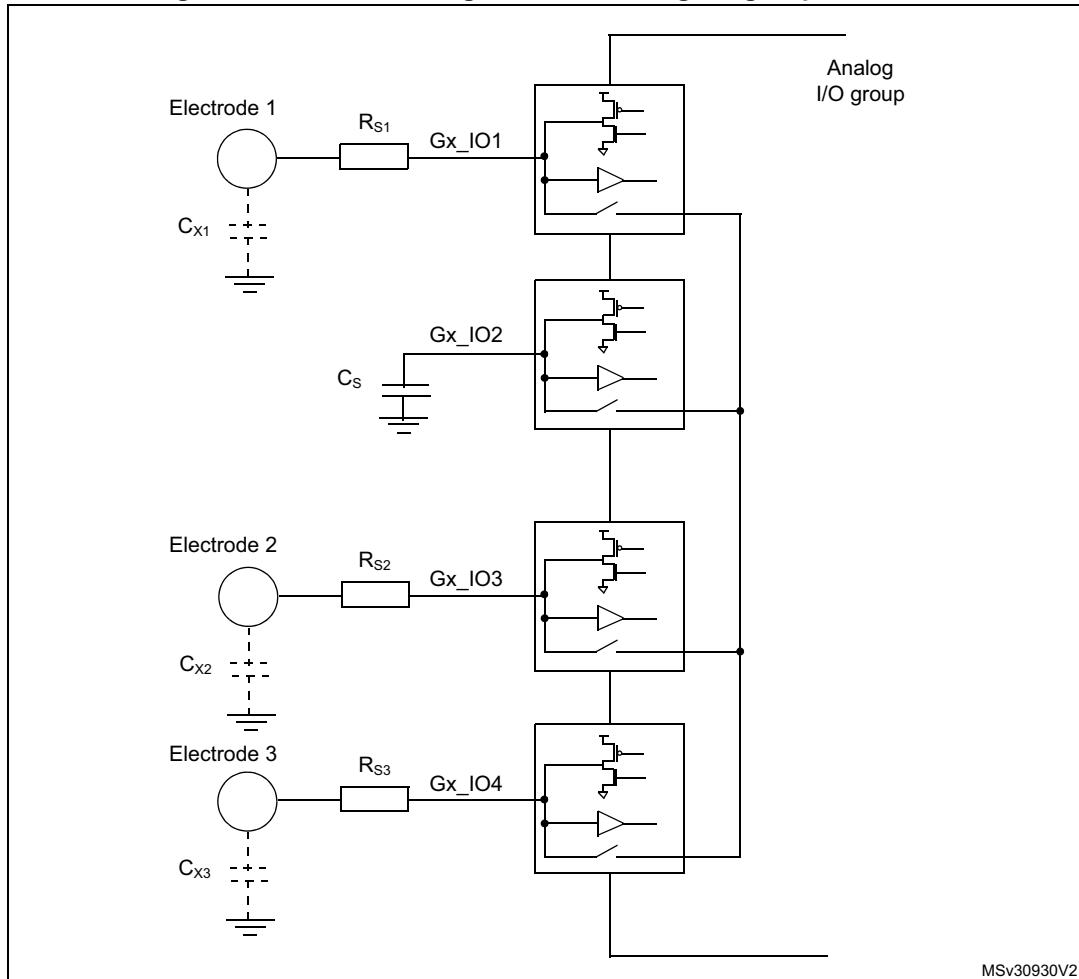
### 20.3.2 Surface charge transfer acquisition overview

The surface charge transfer acquisition is a proven, robust and efficient way to measure a capacitance. It uses a minimum number of external components to operate with a single ended electrode type. This acquisition is designed around an analog I/O group composed of up to four GPIOs (see [Figure 109](#)). Several analog I/O groups are available to allow the acquisition of several capacitive sensing channels simultaneously and to support a larger number of capacitive sensing channels. Within a same analog I/O group, the acquisition of the capacitive sensing channels is sequential.

One of the GPIOs is dedicated to the sampling capacitor  $C_S$ . Only one sampling capacitor I/O per analog I/O group must be enabled at a time.

The remaining GPIOs are dedicated to the electrodes and are commonly called channels. For some specific needs (such as proximity detection), it is possible to simultaneously enable more than one channel per analog I/O group.

Figure 109. Surface charge transfer analog I/O group structure



Note:  $Gx\_IOy$  where  $x$  is the analog I/O group number and  $y$  the GPIO number within the selected group.

The surface charge transfer acquisition principle consists of charging an electrode capacitance ( $C_X$ ) and transferring a part of the accumulated charge into a sampling capacitor ( $C_S$ ). This sequence is repeated until the voltage across  $C_S$  reaches a given threshold ( $V_{IH}$  in our case). The number of charge transfers required to reach the threshold is a direct representation of the size of the electrode capacitance.

[Table 107](#) details the charge transfer acquisition sequence of the capacitive sensing channel 1. States 3 to 7 are repeated until the voltage across  $C_S$  reaches the given threshold. The same sequence applies to the acquisition of the other channels. The electrode serial resistor  $R_S$  improves the ESD immunity of the solution.

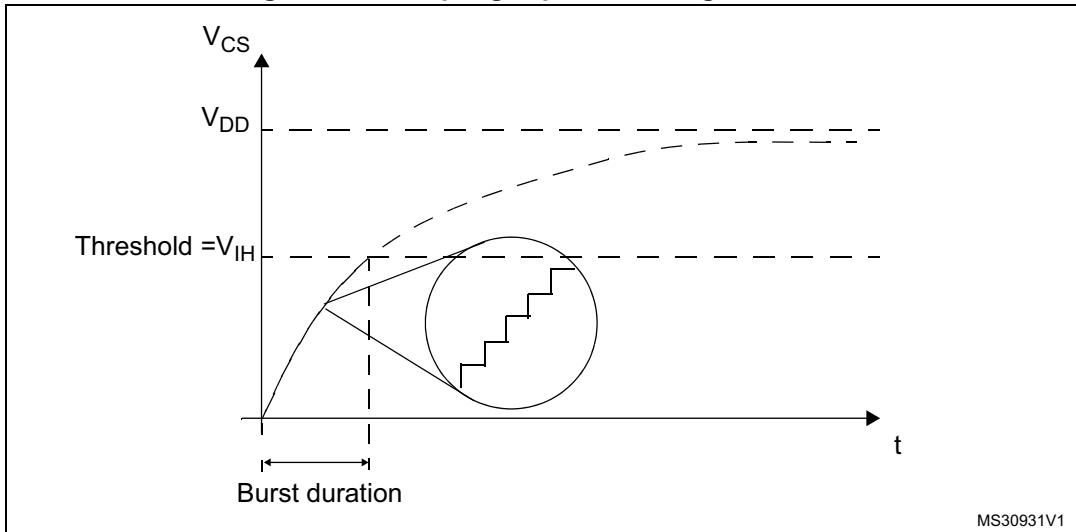
Table 107. Acquisition sequence summary

State	Gx_IO1 (channel)	Gx_IO2 (sampling)	Gx_IO3 (channel)	Gx_IO4 (channel)	State description		
#1	Input floating with analog switch closed	Output open-drain low with analog switch closed	Input floating with analog switch closed		Discharge all $C_X$ and $C_S$		
#2	Input floating				Dead time		
#3	Output push-pull high	Input floating			Charge $C_{X1}$		
#4	Input floating				Dead time		
#5	Input floating with analog switch closed	Input floating			Charge transfer from $C_{X1}$ to $C_S$		
#6	Input floating				Dead time		
#7	Input floating				Measure $C_S$ voltage		

Note:  $Gx\_IOy$  where  $x$  is the analog I/O group number and  $y$  the GPIO number within the selected group.

The voltage variation over the time on the sampling capacitor  $C_S$  is detailed below:

Figure 110. Sampling capacitor voltage variation



### 20.3.3 Reset and clocks

The TSC clock source is the AHB clock (HCLK). Two programmable prescalers are used to generate the pulse generator and the spread spectrum internal clocks:

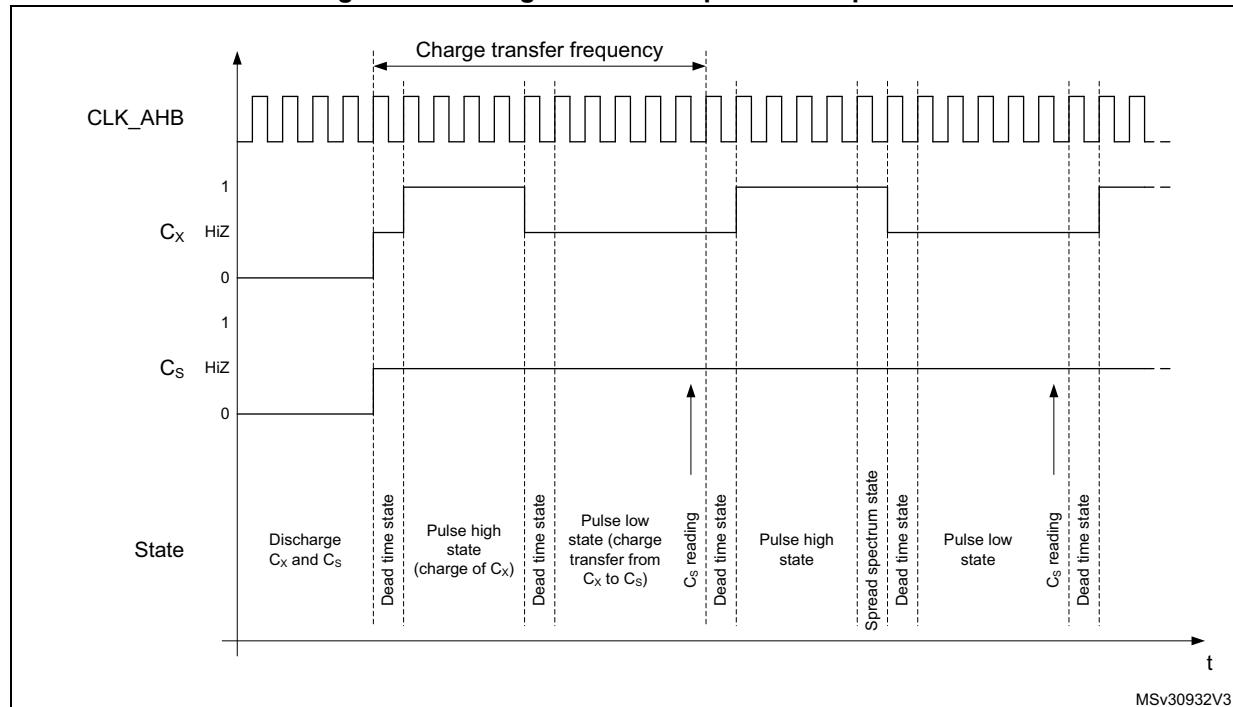
- The pulse generator clock (PGCLK) is defined using the PG\_PSC[2:0] bits of the TSC\_CR register
- The spread spectrum clock (SSCLK) is defined using the SSPSC bit of the TSC\_CR register

The Reset and Clock Controller (RCC) provides dedicated bits to enable the touch sensing controller clock and to reset this peripheral. For more information, refer to [Section 8: Reset and clock control \(RCC\)](#).

### 20.3.4 Charge transfer acquisition sequence

An example of a charge transfer acquisition sequence is detailed in [Figure 111](#).

**Figure 111. Charge transfer acquisition sequence**



For higher flexibility, the charge transfer frequency is fully configurable. Both the pulse high state (charge of  $C_X$ ) and the pulse low state (transfer of charge from  $C_X$  to  $C_S$ ) duration can be defined using the CTPH[3:0] and CTPL[3:0] bits in the TSC\_CR register. The standard range for the pulse high and low states duration is 500 ns to 2  $\mu$ s. To ensure a correct measurement of the electrode capacitance, the pulse high state duration must be set to ensure that  $C_X$  is always fully charged.

A dead time where both the sampling capacitor I/O and the channel I/O are in input floating state is inserted between the pulse high and low states to ensure an optimum charge transfer acquisition sequence. This state duration is 2 periods of HCLK.

At the end of the pulse high state and if the spread spectrum feature is enabled, a variable number of periods of the SSCLK clock are added.

The reading of the sampling capacitor I/O, to determine if the voltage across  $C_S$  has reached the given threshold, is performed at the end of the pulse low state.

**Note:**

*The following TSC control register configurations are forbidden:*

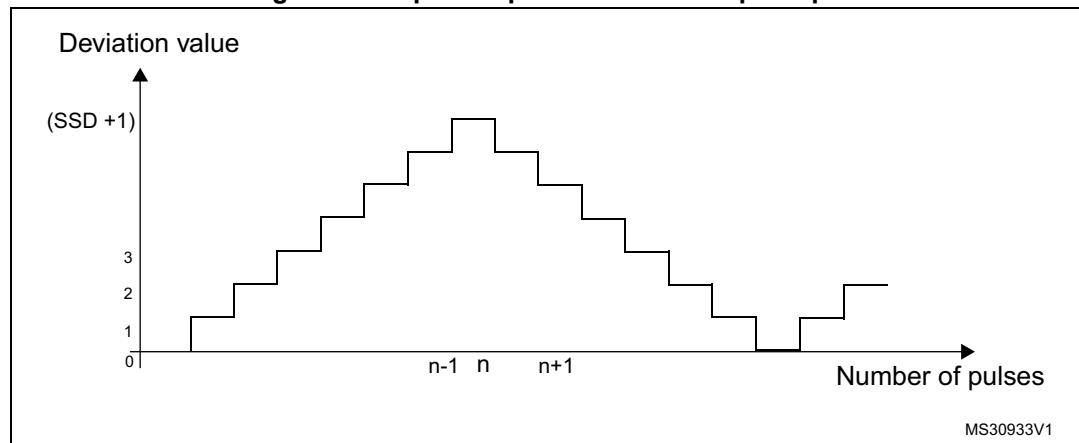
- bits PGPSC are set to '000' and bits CTPL are set to '0000'
- bits PGPSC are set to '000' and bits CTPL are set to '0001'
- bits PGPSC are set to '001' and bits CTPL are set to '0000'

### 20.3.5 Spread spectrum feature

The spread spectrum feature generates a variation of the charge transfer frequency. This is done to improve the robustness of the charge transfer acquisition in noisy environments and also to reduce the induced emission. The maximum frequency variation is in the range of 10% to 50% of the nominal charge transfer period. For instance, for a nominal charge transfer frequency of 250 kHz (4  $\mu$ s), the typical spread spectrum deviation is 10% (400 ns) which leads to a minimum charge transfer frequency of ~227 kHz.

In practice, the spread spectrum consists of adding a variable number of SSCLK periods to the pulse high state using the principle shown below:

**Figure 112. Spread spectrum variation principle**



The table below details the maximum frequency deviation with different HCLK settings:

**Table 108. Spread spectrum deviation versus AHB clock frequency**

$f_{HCLK}$	Spread spectrum step	Maximum spread spectrum deviation
32 MHz	31.25 ns	8206 ns

The spread spectrum feature can be disabled/enabled using the SSE bit in the TSC\_CR register. The frequency deviation is also configurable to accommodate the device HCLK clock frequency and the selected charge transfer frequency through the SSPSC and SSD[6:0] bits in the TSC\_CR register.

### 20.3.6 Max count error

The max count error prevents long acquisition times resulting from a faulty capacitive sensing channel. It consists of specifying a maximum count value for the analog I/O group counters. This maximum count value is specified using the MCV[2:0] bits in the TSC\_CR register. As soon as an acquisition group counter reaches this maximum value, the ongoing acquisition is stopped and the end of acquisition (EOAF bit) and max count error (MCEF bit) flags are both set. An interrupt can also be generated if the corresponding end of acquisition (EOAIE bit) or/and max count error (MCEIE bit) interrupt enable bits are set.

### 20.3.7 Sampling capacitor I/O and channel I/O mode selection

To allow the GPIOs to be controlled by the touch sensing controller, the corresponding alternate function must be enabled through the standard GPIO registers and the GPIOxAFR registers.

The GPIOs modes controlled by the TSC are defined using the TSC\_IOSCR and TSC\_IOCCR register.

When there is no ongoing acquisition, all the I/Os controlled by the touch sensing controller are in default state. While an acquisition is ongoing, only unused I/Os (neither defined as sampling capacitor I/O nor as channel I/O) are in default state. The IODEF bit in the TSC\_CR register defines the configuration of the I/Os which are in default state. The table below summarizes the configuration of the I/O depending on its mode.

**Table 109. I/O state depending on its mode and IODEF bit value**

IODEF bit	Acquisition status	Unused I/O mode	Channel I/O mode	Sampling capacitor I/O mode
0 (output push-pull low)	No	Output push-pull low	Output push-pull low	Output push-pull low
0 (output push-pull low)	Ongoing	Output push-pull low	-	-
1 (input floating)	No	Input floating	Input floating	Input floating
1 (input floating)	Ongoing	Input floating	-	-

#### Unused I/O mode

An unused I/O corresponds to a GPIO controlled by the TSC peripheral but not defined as an electrode I/O nor as a sampling capacitor I/O.

#### Sampling capacitor I/O mode

To allow the control of the sampling capacitor I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output open drain mode and then the corresponding Gx\_IOy bit in the TSC\_IOSCR register must be set.

Only one sampling capacitor per analog I/O group must be enabled at a time.

#### Channel I/O mode

To allow the control of the channel I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output push-pull mode and the corresponding Gx\_IOy bit in the TSC\_IOCCR register must be set.

For proximity detection where a higher equivalent electrode surface is required or to speed-up the acquisition process, it is possible to enable and simultaneously acquire several channels belonging to the same analog I/O group.

#### Note:

*During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC\_IOSCR or TSC\_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.*

### 20.3.8 Acquisition mode

The touch sensing controller offers two acquisition modes:

- Normal acquisition mode: the acquisition starts as soon as the START bit in the TSC\_CR register is set.
- Synchronized acquisition mode: the acquisition is enabled by setting the START bit in the TSC\_CR register but only starts upon the detection of a falling edge or a rising edge and high level on the SYNC input pin. This mode is useful for synchronizing the capacitive sensing channels acquisition with an external signal without additional CPU load.

The GxE bits in the TSC\_IOGCSR registers specify which analog I/O groups are enabled (corresponding counter is counting). The  $C_S$  voltage of a disabled analog I/O group is not monitored and this group does not participate in the triggering of the end of acquisition flag. However, if the disabled analog I/O group contains some channels, they are pulsed.

When the  $C_S$  voltage of an enabled analog I/O group reaches the given threshold, the corresponding GxS bit of the TSC\_IOGCSR register is set. When the acquisition of all enabled analog I/O groups is complete (all GxS bits of all enabled analog I/O groups are set), the EOAF flag in the TSC\_ISR register is set. An interrupt request is generated if the EOAIE bit in the TSC\_IER register is set.

In the case that a max count error is detected, the ongoing acquisition is stopped and both the EOAF and MCEF flags in the TSC\_ISR register are set. Interrupt requests can be generated for both events if the corresponding bits (EOAIE and MCEIE bits of the TSCIER register) are set. Note that when the max count error is detected the remaining GxS bits in the enabled analog I/O groups are not set.

To clear the interrupt flags, the corresponding EOAC and MCEIC bits in the TSC\_ICR register must be set.

The analog I/O group counters are cleared when a new acquisition is started. They are updated with the number of charge transfer cycles generated on the corresponding channel(s) upon the completion of the acquisition.

### 20.3.9 I/O hysteresis and analog switch control

In order to offer a higher flexibility, the touch sensing controller is able to take the control of the Schmitt trigger hysteresis and analog switch of each Gx\_IOy. This control is available whatever the I/O control mode is (controlled by standard GPIO registers or other peripherals) assuming that the touch sensing controller is enabled. This may be useful to perform a different acquisition sequence or for other purposes.

In order to improve the system immunity, the Schmitt trigger hysteresis of the GPIOs controlled by the TSC must be disabled by resetting the corresponding Gx\_IOy bit in the TSC\_IOHCR register.

## 20.4 TSC low-power modes

Table 110. Effect of low-power modes on TSC

Mode	Description
Sleep	No effect. Peripheral interrupts cause the device to exit Sleep mode.
Low power run	No effect.
Low power sleep	No effect. Peripheral interrupts cause the device to exit Low-power sleep mode.
Stop 0 / Stop 1	Peripheral registers content is kept.
Stop 2	
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

## 20.5 TSC interrupts

Table 111. Interrupt control bits

Interrupt event	Enable control bit	Event flag	Clear flag bit	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
End of acquisition	EOAIE	EOAIF	EOAIC	Yes	No	No
Max count error	MCEIE	MCEIF	MCEIC	Yes	No	No

## 20.6 TSC registers

Refer to [Section 1.2 on page 61](#) of the reference manual for a list of abbreviations used in register descriptions.

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

### 20.6.1 TSC control register (TSC\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
CTPH[3:0]				CTPL[3:0]				SSD[6:0]								SSE	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
SSPSC	PGPSC[2:0]				Res.	Res.	Res.	Res.	MCV[2:0]				IODEF	SYNC POL	AM	START	TSCE
rw	rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:28 **CTPH[3:0]**: Charge transfer pulse high

These bits are set and cleared by software. They define the duration of the high state of the charge transfer pulse (charge of  $C_X$ ).

0000: 1x  $t_{PGCLK}$   
0001: 2x  $t_{PGCLK}$   
...

1111: 16x  $t_{PGCLK}$

*Note: These bits must not be modified when an acquisition is ongoing.*

Bits 27:24 **CTPL[3:0]**: Charge transfer pulse low

These bits are set and cleared by software. They define the duration of the low state of the charge transfer pulse (transfer of charge from  $C_X$  to  $C_S$ ).

0000: 1x  $t_{PGCLK}$   
0001: 2x  $t_{PGCLK}$   
...

1111: 16x  $t_{PGCLK}$

*Note: These bits must not be modified when an acquisition is ongoing.*

*Note: Some configurations are forbidden. Refer to the [Section 20.3.4: Charge transfer acquisition sequence](#) for details.*

Bits 23:17 **SSD[6:0]**: Spread spectrum deviation

These bits are set and cleared by software. They define the spread spectrum deviation which consists in adding a variable number of periods of the SSCLK clock to the charge transfer pulse high state.

0000000: 1x  $t_{SSCLK}$   
0000001: 2x  $t_{SSCLK}$   
...

1111111: 128x  $t_{SSCLK}$

*Note: These bits must not be modified when an acquisition is ongoing.*

Bit 16 **SSE**: Spread spectrum enable

This bit is set and cleared by software to enable/disable the spread spectrum feature.

0: Spread spectrum disabled

1: Spread spectrum enabled

*Note: This bit must not be modified when an acquisition is ongoing.*

Bit 15 **SSPSC**: Spread spectrum prescaler

This bit is set and cleared by software. It selects the AHB clock divider used to generate the spread spectrum clock (SSCLK).

0:  $f_{HCLK}$

1:  $f_{HCLK} /2$

*Note: This bit must not be modified when an acquisition is ongoing.*

Bits 14:12 **PGPSC[2:0]**: Pulse generator prescaler

These bits are set and cleared by software. They select the AHB clock divider used to generate the pulse generator clock (PGCLK).

000:  $f_{HCLK}$

001:  $f_{HCLK} /2$

010:  $f_{HCLK} /4$

011:  $f_{HCLK} /8$

100:  $f_{HCLK} /16$

101:  $f_{HCLK} /32$

110:  $f_{HCLK} /64$

111:  $f_{HCLK} /128$

*Note: These bits must not be modified when an acquisition is ongoing.*

*Note: Some configurations are forbidden. Refer to the [Section 20.3.4: Charge transfer acquisition sequence](#) for details.*

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:5 **MCV[2:0]**: Max count value

These bits are set and cleared by software. They define the maximum number of charge transfer pulses that can be generated before a max count error is generated.

000: 255

001: 511

010: 1023

011: 2047

100: 4095

101: 8191

110: 16383

111: reserved

*Note: These bits must not be modified when an acquisition is ongoing.*

Bit 4 **IODEF**: I/O Default mode

This bit is set and cleared by software. It defines the configuration of all the TSC I/Os when there is no ongoing acquisition. When there is an ongoing acquisition, it defines the configuration of all unused I/Os (not defined as sampling capacitor I/O or as channel I/O).

0: I/Os are forced to output push-pull low

1: I/Os are in input floating

*Note: This bit must not be modified when an acquisition is ongoing.*

Bit 3 **SYNCPOL**: Synchronization pin polarity

This bit is set and cleared by software to select the polarity of the synchronization input pin.

0: Falling edge only

1: Rising edge and high level

Bit 2 **AM**: Acquisition mode

This bit is set and cleared by software to select the acquisition mode.

0: Normal acquisition mode (acquisition starts as soon as START bit is set)

1: Synchronized acquisition mode (acquisition starts if START bit is set and when the selected signal is detected on the SYNC input pin)

*Note: This bit must not be modified when an acquisition is ongoing.*

Bit 1 **START**: Start a new acquisition

This bit is set by software to start a new acquisition. It is cleared by hardware as soon as the acquisition is complete or by software to cancel the ongoing acquisition.

0: Acquisition not started

1: Start a new acquisition

Bit 0 **TSCE**: Touch sensing controller enable

This bit is set and cleared by software to enable/disable the touch sensing controller.

0: Touch sensing controller disabled

1: Touch sensing controller enabled

*Note: When the touch sensing controller is disabled, TSC registers settings have no effect.*

## 20.6.2 TSC interrupt enable register (TSC\_IER)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEIE**: Max count error interrupt enable

This bit is set and cleared by software to enable/disable the max count error interrupt.

0: Max count error interrupt disabled

1: Max count error interrupt enabled

Bit 0 **EOAIE**: End of acquisition interrupt enable

This bit is set and cleared by software to enable/disable the end of acquisition interrupt.

0: End of acquisition interrupt disabled

1: End of acquisition interrupt enabled

### 20.6.3 TSC interrupt clear register (TSC\_ICR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

**Bit 1 MCEIC:** Max count error interrupt clear

This bit is set by software to clear the max count error flag and it is cleared by hardware when the flag is reset. Writing a '0' has no effect.

0: No effect

1: Clears the corresponding MCEF of the TSC\_ISR register

**Bit 0 EOAIC:** End of acquisition interrupt clear

This bit is set by software to clear the end of acquisition flag and it is cleared by hardware when the flag is reset. Writing a '0' has no effect.

0: No effect

1: Clears the corresponding EOAF of the TSC\_ISR register

### 20.6.4 TSC interrupt status register (TSC\_ISR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEF**: Max count error flag

This bit is set by hardware as soon as an analog I/O group counter reaches the max count value specified. It is cleared by software writing 1 to the bit MCEIC of the TSC\_ICR register.

0: No max count error (MCE) detected  
1: Max count error (MCE) detected

Bit 0 **EOAF**: End of acquisition flag

This bit is set by hardware when the acquisition of all enabled group is complete (all GxS bits of all enabled analog I/O groups are set or when a max count error is detected). It is cleared by software writing 1 to the bit EOAC of the TSC\_ICR register.

0: Acquisition is ongoing or not started  
1: Acquisition is complete

### 20.6.5 TSC I/O hysteresis control register (TSC\_Iohcr)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw															

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

Bits 27:0 **Gx\_IOy**: Gx\_IOy Schmitt trigger hysteresis mode, x = 8 to 1, y = 4 to 1.

These bits are set and cleared by software to enable/disable the Gx\_IOy Schmitt trigger hysteresis.

0: Gx\_IOy Schmitt trigger hysteresis disabled  
1: Gx\_IOy Schmitt trigger hysteresis enabled

*Note: These bits control the I/O Schmitt trigger hysteresis whatever the I/O control mode is (even if controlled by standard GPIO registers).*

## 20.6.6 TSC I/O analog switch control register (TSC\_IOASCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw															

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

Bits 27:0 **Gx\_IOy**: Gx\_IOy analog switch enable

These bits are set and cleared by software to enable/disable the Gx\_IOy analog switch.

0: Gx\_IOy analog switch disabled (opened)

1: Gx\_IOy analog switch enabled (closed)

*Note: These bits control the I/O analog switch whatever the I/O control mode is (even if controlled by standard GPIO registers).*

## 20.6.7 TSC I/O sampling control register (TSC\_IOSCR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw															

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

Bits 27:0 **Gx\_IOy**: Gx\_IOy sampling mode

These bits are set and cleared by software to configure the Gx\_IOy as a sampling capacitor I/O. Only one I/O per analog I/O group must be defined as sampling capacitor.

0: Gx\_IOy unused

1: Gx\_IOy used as sampling capacitor

*Note: These bits must not be modified when an acquisition is ongoing.*

*During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC\_IOSCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.*

## 20.6.8 TSC I/O channel control register (TSC\_IOCCR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw															

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

Bits 27:0 **Gx\_IOy**: Gx\_IOy channel mode

These bits are set and cleared by software to configure the Gx\_IOy as a channel I/O.

0: Gx\_IOy unused

1: Gx\_IOy used as channel

*Note: These bits must not be modified when an acquisition is ongoing.*

*During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC\_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.*

## 20.6.9 TSC I/O group control status register (TSC\_IGCSR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	G7S	G6S	G5S	G4S	G3S	G2S	G1S								
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	G7E	G6E	G5E	G4E	G3E	G2E	G1E								
									rw						

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

Bits 22:16 **GxS**: Analog I/O group x status

These bits are set by hardware when the acquisition on the corresponding enabled analog I/O group x is complete. They are cleared by hardware when a new acquisition is started.

0: Acquisition on analog I/O group x is ongoing or not started

1: Acquisition on analog I/O group x is complete

*Note: When a max count error is detected the remaining GxS bits of the enabled analog I/O groups are not set.*

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

Bits 6:0 **GxE**: Analog I/O group x enable

These bits are set and cleared by software to enable/disable the acquisition (counter is counting) on the corresponding analog I/O group x.

0: Acquisition on analog I/O group x disabled

1: Acquisition on analog I/O group x enabled

### 20.6.10 TSC I/O group x counter register (TSC\_IOGxCR)

x represents the analog I/O group number.

Address offset: 0x30 + 0x04 \* x, (x = 1 to 7)

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
Res.	Res.	CNT[13:0]													
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 13:0 **CNT[13:0]**: Counter value

These bits represent the number of charge transfer cycles generated on the analog I/O group x to complete its acquisition (voltage across  $C_S$  has reached the threshold).

## 20.6.11 TSC register map

**Table 112. TSC register map and reset values**

Table 112. TSC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x003C	<b>TSC_IOG3CR</b>	Res.																															
	Reset value	Res.	0	0	0	0	0	0	0	0	0	0	0																				
0x0040	<b>TSC_IOG4CR</b>	Res.																															
	Reset value	Res.	0	0	0	0	0	0	0	0	0	0	0																				
0x0044	<b>TSC_IOG5CR</b>	Res.																															
	Reset value	Res.	0	0	0	0	0	0	0	0	0	0	0																				
0x0048	<b>TSC_IOG6CR</b>	Res.																															
	Reset value	Res.	0	0	0	0	0	0	0	0	0	0	0																				
0x004C	<b>TSC_IOG7CR</b>	Res.																															
	Reset value	Res.	0	0	0	0	0	0	0	0	0	0	0																				

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 21 True random number generator (RNG)

### 21.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG can be used to construct a NIST compliant Deterministic Random Bit Generator (DRBG), acting as a live entropy source.

The RNG true random number generator has been tested using German BSI statistical tests of AIS-31 (T0 to T8).

### 21.2 RNG main features

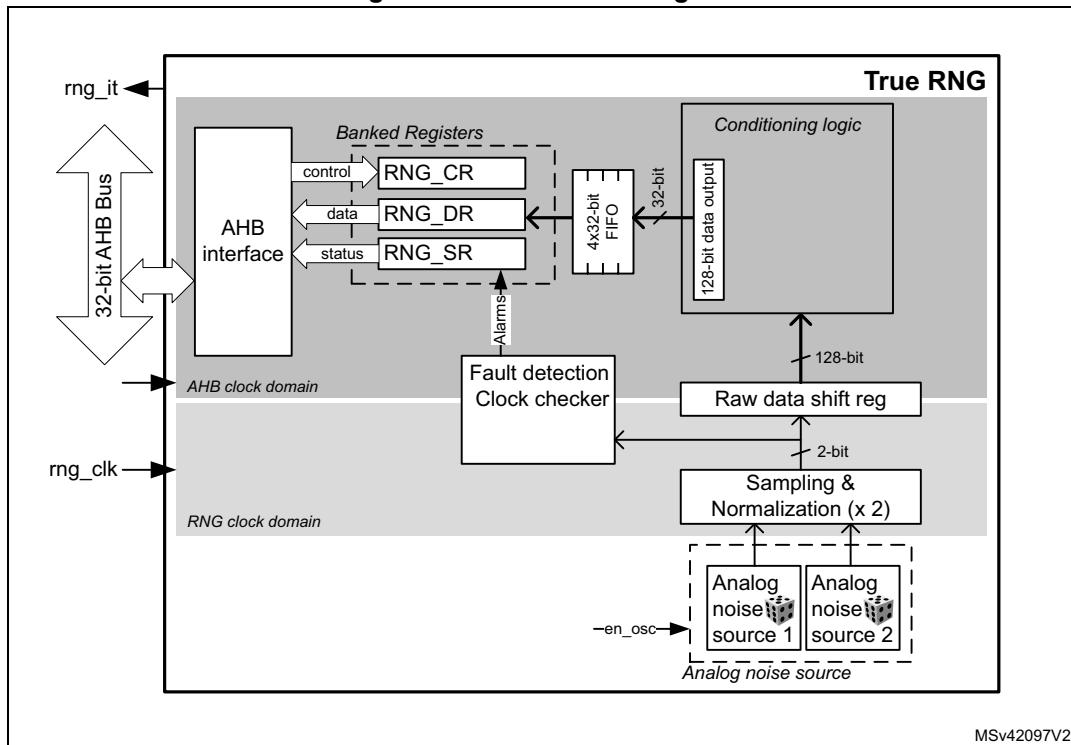
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source processed by a high quality conditioning stage.
- In the NIST configuration, it produces four 32-bit random samples every  $16 \times \frac{f_{AHB}}{f_{RNG}}$  AHB clock cycles, if value is higher than 213 cycles (213 cycles otherwise).
- It allows embedded continuous basic health tests with associated error management
  - Includes too low sampling clock detection and repetition count tests.
- It can be disabled to reduce power consumption.
- It has an AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

## 21.3 RNG functional description

### 21.3.1 RNG block diagram

Figure 113 shows the RNG block diagram.

Figure 113. RNG block diagram



MSv42097V2

### 21.3.2 RNG internal signals

Table 113 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 113. RNG internal input/output signals

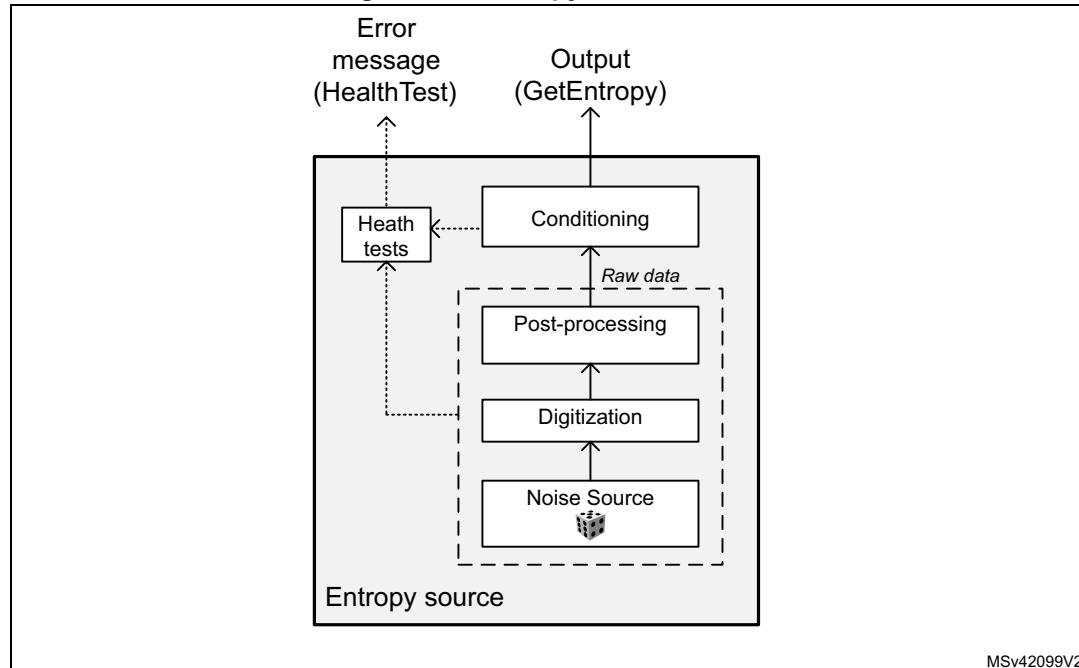
Signal name	Signal type	Description
<code>rng_it</code>	Digital output	RNG global interrupt request
<code>rng_hclk</code>	Digital input	AHB clock
<code>rng_clk</code>	Digital input	RNG dedicated clock, asynchronous to <code>rng_hclk</code>

### 21.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals. Within its boundary the RNG implements the entropy source model pictured on [Figure 114](#).

It includes an analog noise source, a digitization stage with post-processing, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

**Figure 114. Entropy source model**



The components pictured above are detailed hereafter.

#### Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. It is composed of:

- Two analog noise sources, each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in [Section 21.3.8: RNG low-power usage](#).
- A sampling stage of these outputs clocked by a dedicated clock input (rng\_clk), delivering a 2-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng\_hclk).

**Note:** *In [Section 21.6: RNG entropy source validation](#) recommended RNG clock frequencies are given.*

## Post processing

The sample values obtained from a true random noise source consist of 2-bit bitstrings. Because this noise source output is biased, the RNG implements a post-processing component that reduces that bias to a tolerable level.

More specifically, for each of the two noise source bits the RNG takes half of the bits from the sampled noise source, and half of the bits from inverted sampled noise source. Thus, if the source generates more '1' than '0' (or the opposite), it is filtered

## Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit).

Also note that post-processing computations are triggered when at least 32 bits of raw data are received and when output FIFO needs a refill. Thus the RNG output entropy is maximum when the RNG 128-bit FIFO is emptied by application after 64 RNG clock cycles.

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in [Section 21.5: RNG processing time](#).

The conditioning component is clocked by the faster AHB clock.

## Output buffer

A data output buffer can store up to four 32-bit words which have been output from the conditioning component. When four words have been read from the output FIFO through the RNG\_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register 213 AHB clock cycles later.

Whenever a random number is available through the RNG\_DR register the DRDY flag transitions from 0 to 1. This flag remains high until output buffer becomes empty after reading four words from the RNG\_DR register.

*Note:*

*When interrupts are enabled an interrupt is generated when this data ready flag transitions from 0 to 1. Interrupt is then cleared automatically by the RNG as explained above.*

## Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features.

1. Continuous health tests, running indefinitely on the output of the noise source
  - Repetition count test, flagging an error when:
    - a) One of the noise source has provided more than 64 consecutive bits at a constant value ("0" or "1"), or more than 32 consecutive occurrence of two bits patterns ("01" or "10")
    - b) Both noise sources have delivered more than 32 consecutive bits at a constant value ("0" or "1"), or more than 16 consecutive occurrence of two bits patterns ("01" or "10")
  - 2. Vendor specific continuous test
    - Real-time "too slow" sampling clock detector, flagging an error when one RNG clock cycle is smaller than AHB clock cycle divided by 32.

The CECS and SECS status bits in the RNG\_SR register indicate when an error condition is detected, as detailed in [Section 21.3.7: Error management](#).

*Note:* An interrupt can be generated when an error is detected.

## 21.3.4 RNG initialization

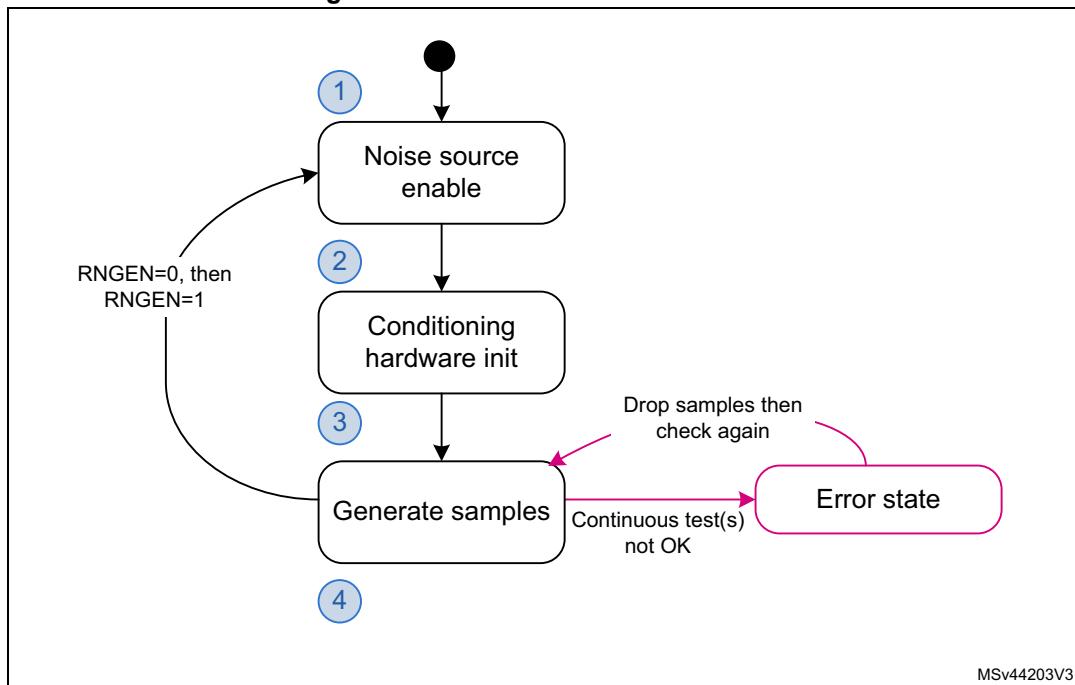
The RNG simplified state machine is pictured on [Figure 115](#).

After enabling the RNG (RNGEN = 1 in RNG\_CR) the following chain of events occurs:

1. The analog noise source is enabled, and logic immediately starts sampling the analog output, filling 128-bit conditioning shift register
2. The conditioning logic is enabled and post-processing context is initialized using two 128 noise source bits.
3. The conditioning stage internal input data buffer is filled again with 128-bit and one conditioning round is performed. The output buffer is then filled with post processing result.
4. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in [Section 21.5: RNG processing time](#).

**Figure 115. RNG initialization overview**



## 21.3.5 RNG operation

### Normal operations

To run the RNG using interrupts, the following steps are recommended:

1. Enable the interrupts by setting the IE bit in the RNG\_CR register. At the same time enable the RNG by setting the bit RNGEN=1.
2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore at each interrupt, check that:
  - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG\_SR register.
  - A random number is ready. The DRDY bit must be set to 1 in the RNG\_SR register.
  - If above two conditions are true the content of the RNG\_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of above conditions are false, the RNG\_DR register must not be read. If an error occurred error recovery sequence described in [Section 21.3.7](#) must be used.

To run the RNG in polling mode following steps are recommended:

1. Enable the random number generation by setting the RNGEN bit to “1” in the RNG\_CR register.
2. Read the RNG\_SR register and check that:
  - No error occurred (the SEIS and CEIS bits must be set to 0)
  - A random number is ready (the DRDY bit must be set to 1)
3. If above conditions are true read the content of the RNG\_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of above conditions are false, the RNG\_DR register must not be read. If an error occurred error recovery sequence described in [Section 21.3.7](#) must be used.

**Note:**

*When data is not ready (DRDY = 0) RNG\_DR returns zero.*

*It is recommended to always verify that RNG\_DR is different from zero. Because when it is the case a seed error occurred between RNG\_SR polling and RND\_DR output reading (rare event).*

### Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in [Section 21.3.8: RNG low-power usage](#).

### Software post-processing

If a NIST approved DRBG with 128 bits of security strength is required an approved random generator software must be built around the RNG true random number generator.

Built-in health check functions are described in [Section 21.3.3: Random number generation](#).

## 21.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock is used for noise source sampling. Recommended clock configurations are detailed in [Section 21.6: RNG entropy source validation](#).

**Note:**

*When the CED bit in the RNG\_CR register is set to 0, the RNG clock frequency **must be higher** than AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG\_SR register).*

See [Section 21.3.1: RNG block diagram](#) for details (AHB and RNG clock domains).

## 21.3.7 Error management

In parallel to random number generation an health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

### Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see

*Section 21.3.6: RNG clocking*) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when clocking condition is normal.

*Note:* The clock error has no impact on generated random numbers, that is the application can still read RNG\_DR register.

*CEIS is set only when CECS is set to 1 by RNG.*

### Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred. If a value is available in the RNG\_DR register, it must not be used as it may not have enough entropy. If the error was detected during the initialization phase the whole initialization sequence is automatically restarted by the RNG.

The following sequence must be used to fully recover from a seed error after the RNG initialization:

1. Clear the SEIS bit by writing it to "0".
2. Read out 12 words from the RNG\_DR register, and discard each of them in order to clean the pipeline.
3. Confirm that SEIS is still cleared. Random number generation is back to normal.

## 21.3.8 RNG low-power usage

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG\_CR register. As the post-processing logic and the output buffer remain operational while RNGEN = 0 following features are available to software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG\_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG\_DR register. If it is not the case the RNG must be re-enabled by the application until at least 32 new bits are collected from the noise source and a complete conditioning round is done. It corresponds to 16 RNG clock cycles to sample new bits, and 216 AHB clock cycles to run a conditioning round.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG\_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when RNGEN bit is set to 1.

## 21.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see [Section 21.3.7: Error management](#)
- Clock error, see [Section 21.3.7: Error management](#)

Dedicated interrupt enable control bits are available as shown in [Table 114](#).

**Table 114. RNG interrupt requests**

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write 0 to SEIS
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG\_CR register. The status of the individual interrupt sources can be read from the RNG\_SR register.

*Note:* *Interrupts are generated only when RNG is enabled.*

## 21.5 RNG processing time

The conditioning stage can produce four 32-bit random numbers every  $16 \times \frac{f_{AHB}}{f_{RNG}}$  clock cycles, if the value is higher than 213 cycles (213 cycles otherwise).

More time is needed for the first set of random numbers after the device exits reset (see [Section 21.3.4: RNG initialization](#)). Indeed, after enabling the RNG for the first time, random data is first available after either:

- 128 RNG clock cycles + 426 AHB cycles, if  $f_{AHB} < f_{threshold}$
- 192 RNG clock cycles + 213 AHB cycles, if  $f_{AHB} \geq f_{threshold}$

With  $f_{threshold} = (213 \times f_{RNG}) / 64$

## 21.6 RNG entropy source validation

### 21.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using German BSI AIS-31 statistical tests (T0 to T8).

### 21.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock  $rng\_clk = 48$  MHz (CED bit = '0' in RNG\_CR register) and  $rng\_clk = 400$  kHz (CED bit = '1' in RNG\_CR register).

## 21.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

### 21.7.1 RNG control register (RNG\_CR)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

Bit 5 **CED**: Clock error detection

- 0: Clock error detection is enable
- 1: Clock error detection is disable

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED the RNG must be disabled.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE**: Interrupt Enable

- 0: RNG Interrupt is disabled
- 1: RNG Interrupt is enabled. An interrupt is pending as soon as DRDY = 1, SEIS = 1 or CEIS = 1 in the RNG\_SR register.

Bit 2 **RNGEN**: True random number generator enable

- 0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.
- 1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.

## 21.7.2 RNG status register (RNG\_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY								
									rc_w0	rc_w0			r	r	r

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

### Bit 6 **SEIS**: Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0. Writing 1 has no effect.

0: No faulty sequence detected

1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG\_CR register.

### Bit 5 **CEIS**: Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct (fRNGCLK > fHCLK/32)

1: The RNG is detected too slow (fRNGCLK < fHCLK/32)

An interrupt is pending if IE = 1 in the RNG\_CR register.

Bits 4:3 Reserved, must be kept at reset value.

### Bit 2 **SECS**: Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

- One of the noise source has provided more than 64 consecutive bits at a constant value ("0" or "1"), or more than 32 consecutive occurrence of two bit patterns ("01" or "10")
- Both noise sources have delivered more than 32 consecutive bits at a constant value ("0" or "1"), or more than 16 consecutive occurrence of two bit patterns ("01" or "10")

### Bit 1 **CECS**: Clock error current status

0: The RNG clock is correct (fRNGCLK > fHCLK/32). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow (fRNGCLK < fHCLK/32).

*Note: CECS bit is valid only if the CED bit in the RNG\_CR register is set to 0.*

### Bit 0 **DRDY**: Data Ready

0: The RNG\_DR register is not yet valid, no random data is available.

1: The RNG\_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG\_DR register), this bit returns to 0 until a new random value is generated.

*Note: The DRDY bit can rise when the peripheral is disabled (RNGEN = 0 in the RNG\_CR register).*

If IE=1 in the RNG\_CR register, an interrupt is generated when DRDY = 1.

### 21.7.3 RNG data register (RNG\_DR)

Address offset: 0x008

Reset value: 0x0000 0000

The RNG\_DR register is a read-only register that delivers a 32-bit random value when read. After being read this register delivers a new random value after 216 periods of AHB clock if the output FIFO is empty.

The content of this register is valid when DRDY = 1 and value is not 0x0, even if RNGEN = 0.

Bits 31:0 **RNDATA[31:0]**: Random data

32-bit random data which are valid when DRDY = 1. When DRDY = 0 RNDATA value is zero.

When DRDY is set, it is recommended to always verify that RNG\_DR is different from zero. Because when it is the case a seed error occurred between RNG\_SR polling and RND\_DR output reading (rare event).

#### 21.7.4 RNG register map

**Table 115. RNG register map and reset map**

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

## 22 AES hardware accelerator (AES)

### 22.1 Introduction

The AES hardware accelerator (AES) encrypts or decrypts data, using an algorithm and implementation fully compliant with the advanced encryption standard (AES) defined in Federal information processing standards (FIPS) publication 197.

The peripheral supports CTR, GCM, GMAC, CCM, ECB, and CBC chaining modes for key sizes of 128 or 256 bits.

AES is an AMBA AHB slave peripheral accessible through 32-bit single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels required).

### 22.2 AES main features

- Compliance with NIST “Advanced encryption standard (AES), FIPS publication 197” from November 2001
- 128-bit data block processing
- Support for cipher key lengths of 128-bit and 256-bit
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block of data with, respectively, 128-bit or 256-bit key
- Integrated round key scheduler to compute the last round key for ECB/CBC decryption
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only
- 256-bit register for storing the cryptographic key (eight 32-bit registers)
- 128-bit register for storing initialization vector (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control with support of single-transfer direct memory access (DMA) using two channels (one for incoming data, one for processed data)
- Data-swapping logic to support 1-, 8-, 16- or 32-bit data
- Possibility for software to suspend a message if AES needs to process another message with a higher priority, then resume the original message

## 22.3 AES implementation

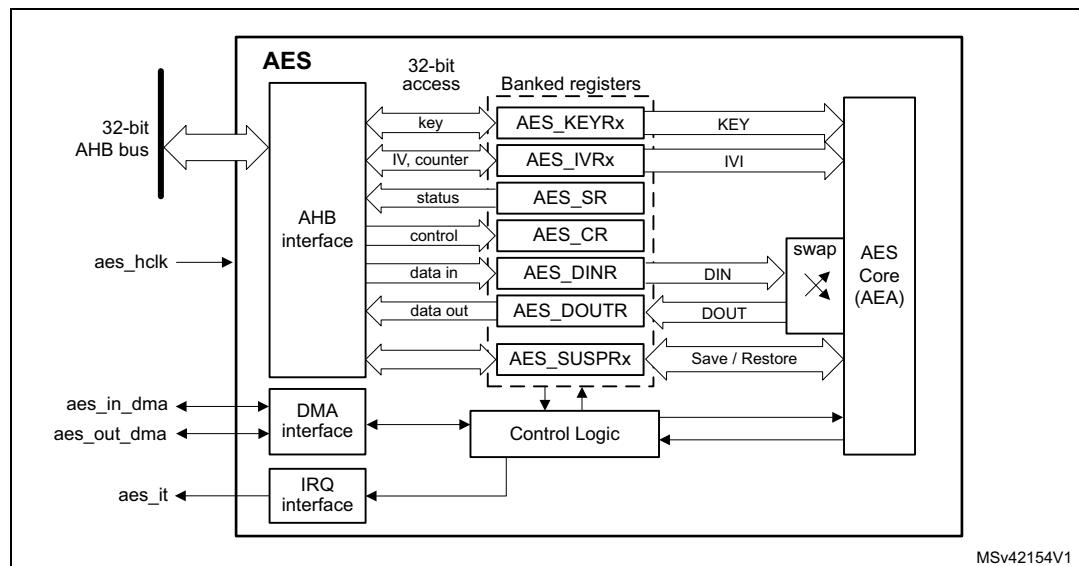
The devices have two AES peripherals, AES1 and AES2.

## 22.4 AES functional description

### 22.4.1 AES block diagram

*Figure 116* shows the block diagram of AES.

Figure 116. AES block diagram



### 22.4.2 AES internal signals

*Table 116* describes the user relevant internal signals interfacing the AES peripheral.

Table 116. AES internal input/output signals

Signal name	Signal type	Description
<b>aes_hclk</b>	Input	AHB bus clock
<b>aes_it</b>	Output	AES interrupt request
<b>aes_in_dma</b>	Input/Output	Input DMA single request/acknowledge
<b>aes_out_dma</b>	Input/Output	Output DMA single request/acknowledge

### 22.4.3 AES cryptographic core

#### Overview

The AES cryptographic core consists of the following components:

- AES core algorithm (AEA)
- multiplier over a binary Galois field (GF2mul)
- key input
- initialization vector (IV) input
- chaining algorithm logic (XOR, feedback/counter, mask)

The AES core works on 128-bit data blocks (four words) with 128-bit or 256-bit key length. Depending on the chaining mode, the AES requires zero or one 128-bit initialization vector IV.

The AES features the following modes of operation:

- **Mode 1:**  
Plaintext encryption using a key stored in the AES\_KEYRx registers
- **Mode 2:**  
ECB or CBC decryption key preparation. It must be used prior to selecting Mode 3 with ECB or CBC chaining modes. The key prepared for decryption is stored automatically in the AES\_KEYRx registers. Now the AES peripheral is ready to switch to Mode 3 for executing data decryption.
- **Mode 3:**  
Ciphertext decryption using a key stored in the AES\_KEYRx registers. When ECB and CBC chaining modes are selected, the key must be prepared beforehand, through Mode 2.
- **Mode 4:**  
ECB or CBC ciphertext single decryption using the key stored in the AES\_KEYRx registers (the initial key is derived automatically).

*Note:* Mode 2 and mode 4 are only used when performing ECB and CBC decryption.

*When Mode 4 is selected only one decryption can be done, therefore usage of Mode 2 and Mode 3 is recommended instead.*

The operating mode is selected by programming the MODE[1:0] bitfield of the AES\_CR register. It may be done only when the AES peripheral is disabled.

#### Typical data processing

Typical usage of the AES is described in [Section 22.4.4: AES procedure to perform a cipher operation on page 593](#).

*Note:* The outputs of the intermediate AEA stages are never revealed outside the cryptographic boundary, with the exclusion of the IVI bitfield.

## Chaining modes

The following chaining modes are supported by AES, selected through the CHMOD[2:0] bitfield of the AES\_CR register:

- Electronic code book (ECB)
- Cipher block chaining (CBC)
- Counter (CTR)
- Galois counter mode (GCM)
- Galois message authentication code (GMAC)
- Counter with CBC-MAC (CCM)

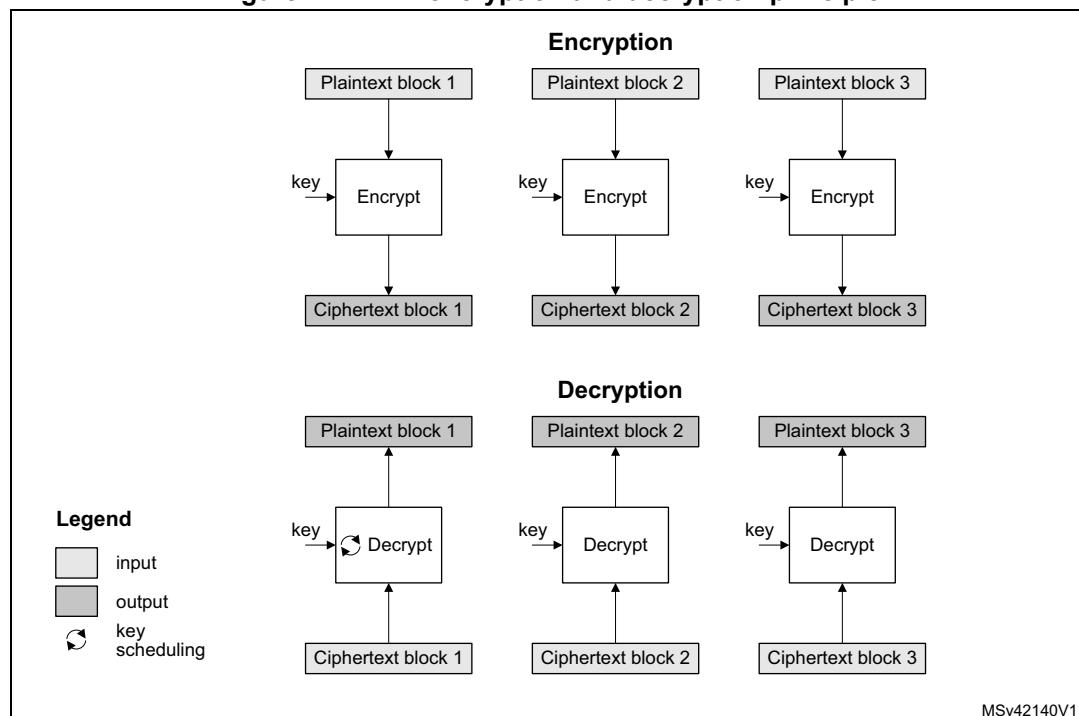
*Note:* The chaining mode may be changed only when AES is disabled (bit EN of the AES\_CR register cleared).

Principle of each AES chaining mode is provided in the following subsections.

Detailed information is in dedicated sections, starting from [Section 22.4.8: AES basic chaining modes \(ECB, CBC\)](#).

### Electronic codebook (ECB) mode

**Figure 117. ECB encryption and decryption principle**

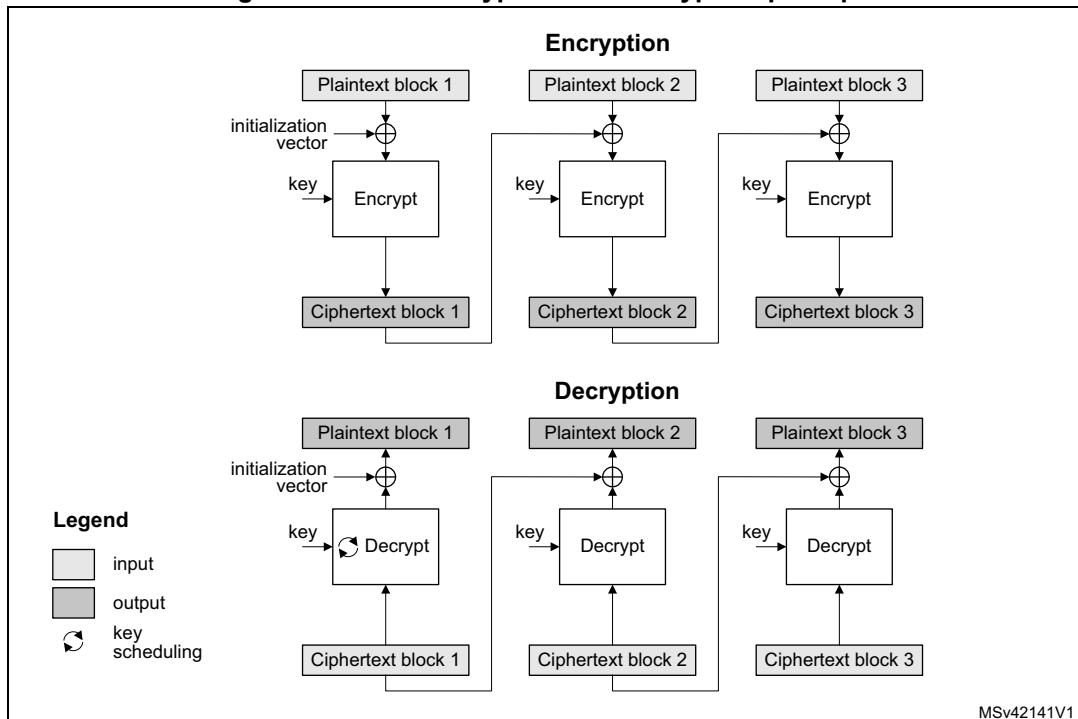


ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

*Note:* For decryption, a special key scheduling is required before processing the first block.

### Cipher block chaining (CBC) mode

Figure 118. CBC encryption and decryption principle

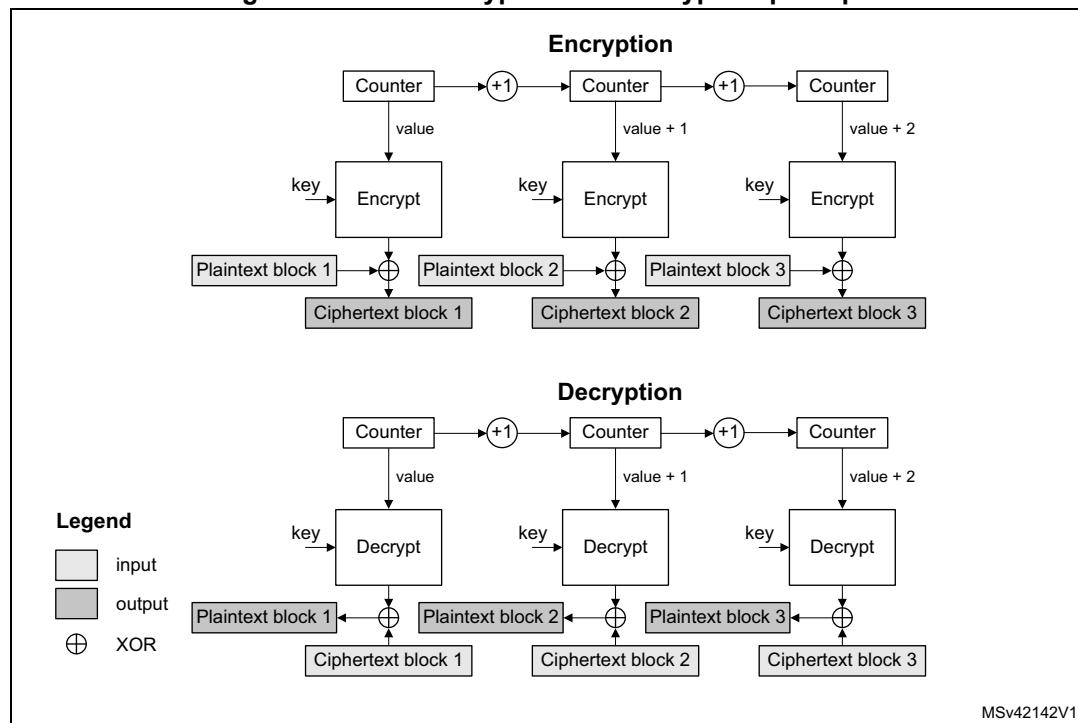


In CBC mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

*Note:* For decryption, a special key scheduling is required before processing the first block.

## Counter (CTR) mode

Figure 119. CTR encryption and decryption principle

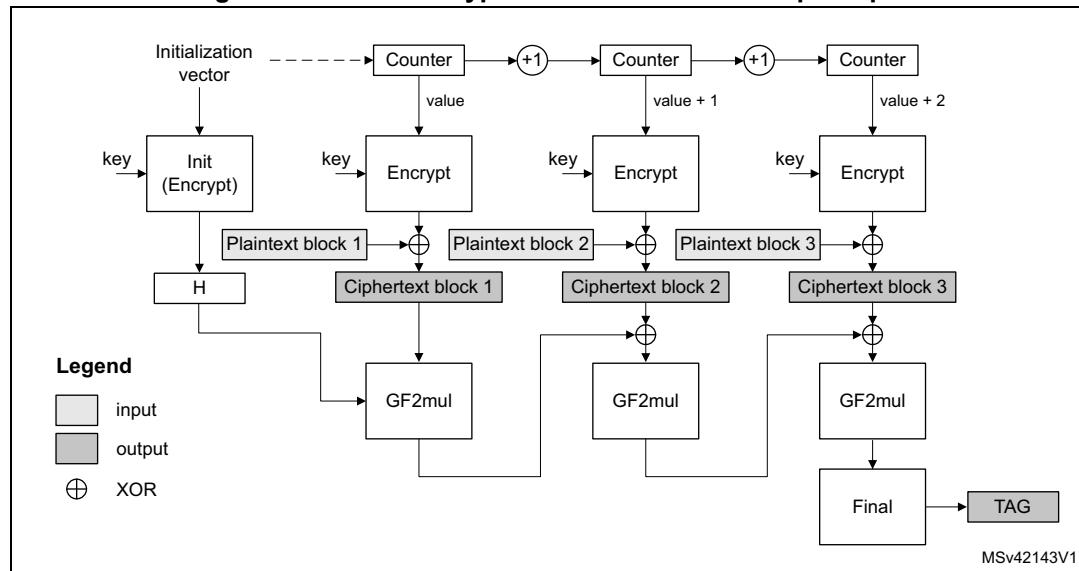


The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext as specified in NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation*.

**Note:** *Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption, since in this chaining scheme the AES core is always used in encryption mode for producing the key stream, or counter blocks.*

### Galois/counter mode (GCM)

Figure 120. GCM encryption and authentication principle

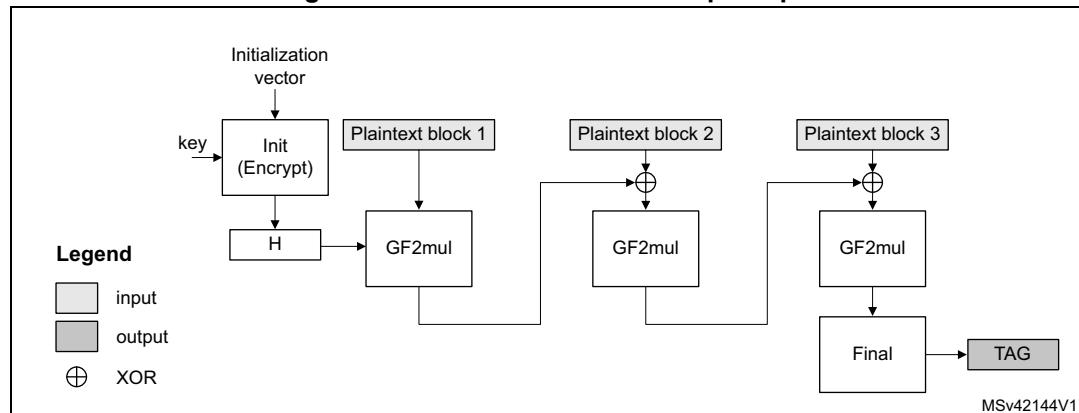


In Galois/counter mode (GCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in *NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. It requires an initial value and a particular 128-bit block at the end of the message.

### Galois message authentication code (GMAC) principle

Figure 121. GMAC authentication principle

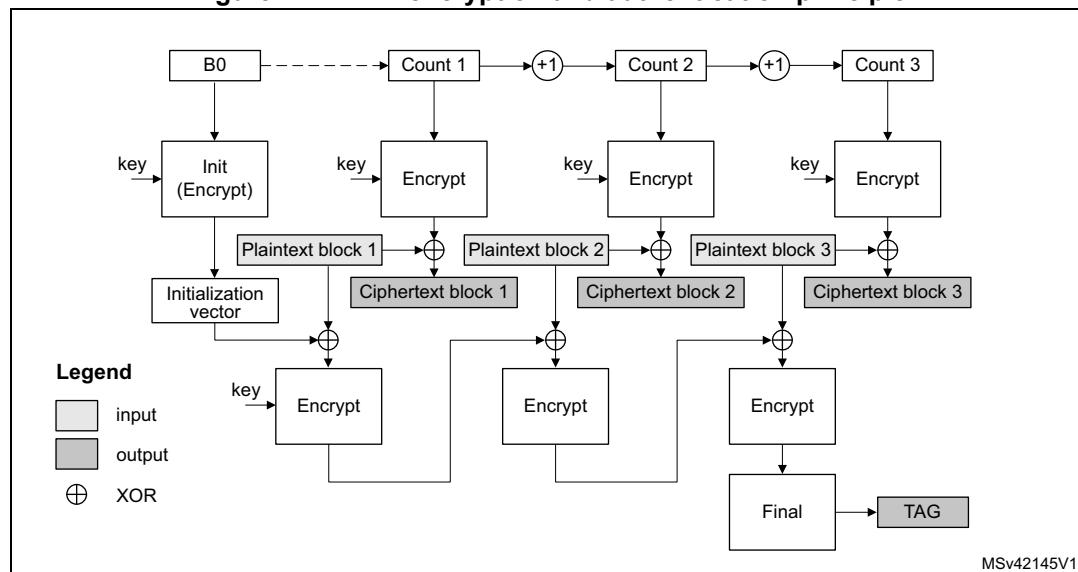


Galois message authentication code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in *NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload).

### Counter with CBC-MAC (CCM) principle

Figure 122. CCM encryption and authentication principle



In Counter with cipher block chaining-message authentication code (CCM) mode, the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and the corresponding MAC (also known as tag). It is described by NIST in *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

CCM mode is based on AES in counter mode for confidentiality and it uses CBC for computing the message authentication code. It requires an initial value.

Like GCM, the CCM chaining mode can be applied on a message composed only by plaintext authenticated data (that is, only header, no payload). Note that this way of using CCM is not called CMAC (it is not similar to GCM/GMAC), and its use is not recommended by NIST.

#### 22.4.4 AES procedure to perform a cipher operation

##### Introduction

A typical cipher operation is explained below. Detailed information is provided in sections starting from [Section 22.4.8: AES basic chaining modes \(ECB, CBC\)](#).

## Initialization of AES

To initialize AES, first disable it by clearing the EN bit of the AES\_CR register. Then perform the following steps in any order:

- Configure the AES mode, by programming the MODE[1:0] bitfield of the AES\_CR register.
  - For encryption, select Mode 1 (MODE[1:0] = 00).
  - For decryption, select Mode 3 (MODE[1:0] = 10), unless ECB or CBC chaining modes are used. In this latter case, perform an initial key derivation of the encryption key, as described in [Section 22.4.5: AES decryption round key preparation](#).
- Select the chaining mode, by programming the CHMOD[2:0] bitfield of the AES\_CR register.
- Configure the data type (1-, 8-, 16- or 32-bit), with the DATATYPE[1:0] bitfield in the AES\_CR register.
- When it is required (for example in CBC or CTR chaining modes), write the initialization vector into the AES\_IVRx registers.
- Configure the key size (128-bit or 256-bit), with the KEYSIZE bitfield of the AES\_CR register.
- Write a symmetric key into the AES\_KEYRx registers (4 or 8 registers depending on the key size).

## Data append

This section describes different ways of appending data for processing, where the size of data to process is not a multiple of 128 bits.

For ECB or CBC mode, refer to [Section 22.4.6: AES ciphertext stealing and data padding](#). The last block management in these cases is more complex than in the sequence described in this section.

### Data append through polling

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
2. Repeat the following sub-sequence until the payload is entirely processed:
  - a) Write four input data words into the AES\_DINR register.
  - b) Wait until the status flag CCF is set in the AES\_SR, then read the four data words from the AES\_DOUTR register.
  - c) Clear the CCF flag, by setting the CCFC bit of the AES\_CR register.
  - d) If the data block just processed is the second-last block of the message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros and, in case of GCM payload encryption or CCM payload decryption, specify the number of non-valid bytes, using the NPBLB bitfield of the AES\_CR register, for AES to compute a correct tag;
3. As it is the last block, discard the data that is not part of the data, then disable the AES peripheral by clearing the EN bit of the AES\_CR register.

**Note:** *Up to three wait cycles are automatically inserted between two consecutive writes to the AES\_DINR register, to allow sending the key to the AES processor.*

*NPBLB bits are not used in header phase of GCM, GMAC and CCM chaining modes.*

### Data append using interrupt

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES by setting the CCFIE bit of the AES\_CR register.
2. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
3. Write first four input data words into the AES\_DINR register.
4. Handle the data in the AES interrupt service routine, upon interrupt:
  - a) Read four output data words from the AES\_DOUTR register.
  - b) Clear the CCF flag and thus the pending interrupt, by setting the CCFC bit of the AES\_CR register.
  - c) If the data block just processed is the second-last block of a message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros and, in case of GCM payload encryption or CCM payload decryption, specify the number of non-valid bytes, using the NPBLB bitfield of the AES\_CR register, for AES to compute a correct tag;. Then proceed with point 4e).
  - d) If the data block just processed is the last block of the message, discard the data that is not part of the data, then disable the AES peripheral by clearing the EN bit of the AES\_CR register and quit the interrupt service routine.
  - e) Write next four input data words into the AES\_DINR register and quit the interrupt service routine.

**Note:** *AES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two AES computations. NPBLB bits are not used in header phase of GCM, GMAC and CCM chaining modes.*

### Data append using DMA

With this method, all the transfers and processing are managed by DMA and AES. To use the method, proceed as follows:

1. Prepare the last four-word data block (if the data to process does not fill it completely), by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the AES peripheral input and the processed data from the AES peripheral output to the memory, as described in [Section 22.4.16: AES DMA interface](#). Configure the DMA controller so as to generate an interrupt on transfer completion. In case of GCM payload encryption or CCM payload decryption, DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in [Data append through polling](#) must be used instead for this last block, because NPBLB bits must be setup before processing the block, for AES to compute a correct tag.
3. Enable the AES peripheral by setting the EN bit of the AES\_CR register
4. Enable DMA requests by setting the DMAINEN and DMAOUTEN bits of the AES\_CR register.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

**Note:** *The CCF flag has no use with this method, because the reading of the AES\_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.*

*NPBLB bits are not used in header phase of GCM, GMAC, and CCM chaining modes.*

### 22.4.5 AES decryption round key preparation

Internal key schedule is used to generate AES round keys. In AES encryption, the round 0 key is the one stored in the key registers. AES decryption must start using the last round key. As the encryption key is stored in memory, a special key scheduling must be performed to obtain the decryption key. This key scheduling is only required for AES decryption in ECB and CBC modes.

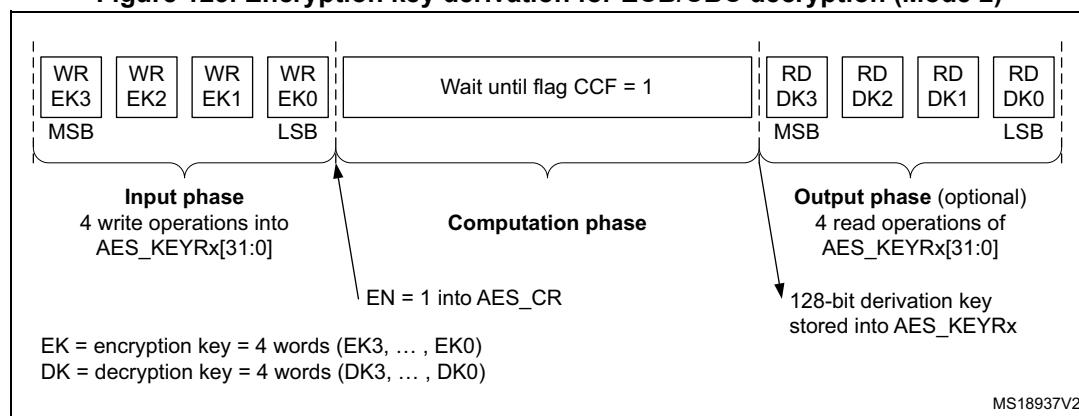
Recommended method is to select the Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES\_CR (key process only), then proceed with the decryption by setting MODE[1:0] to 10 (Mode 3, decryption only). Mode 2 usage is described below:

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES\_CR. The CHMOD[2:0] bitfield is not significant in this case because this key derivation mode is independent of the chaining algorithm selected.
3. Set key length to 128 or 256 bits, via KEYSIZE bit of AES\_CR register.
4. Write the AES\_KEYRx registers (128 or 256 bits) with encryption key, as shown in [Figure 123](#). Writes to the AES\_IVRx registers have no effect.
5. Enable the AES peripheral, by setting the EN bit of the AES\_CR register.
6. Wait until the CCF flag is set in the AES\_SR register.
7. Clear the CCF flag. Derived key is available in AES core, ready to use for decryption. Application can also read the AES\_KEYRx register to obtain the derived key if needed, as shown in [Figure 123](#) (the processed key is loaded automatically into the AES\_KEYRx registers).

**Note:** *The AES is disabled by hardware when the derivation key is available.*

*To restart a derivation key computation, repeat steps 4, 5, 6, and 7.*

**Figure 123. Encryption key derivation for ECB/CBC decryption (Mode 2)**



If the software stores the initial key prepared for decryption, it is enough to do the key schedule operation only once for all the data to be decrypted with a given cipher key.

**Note:** *The operation of the key preparation lasts 59 or 82 clock cycles, depending on the key size (128- or 256-bit).*

## 22.4.6 AES ciphertext stealing and data padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (128 bits), ciphertext stealing techniques are used, such as those described in *NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since the AES peripheral does not support such techniques, the application must complete the last block of input data using data from the second last block.

**Note:** *Ciphertext stealing techniques are not documented in this reference manual.*

Similarly, when AES is used in other modes than ECB or CBC, an incomplete input data block (that is, block with input data shorter than 128 bits) must be padded with zeros prior to encryption (that is, extra bits must be appended to the trailing end of the data string). After decryption, the extra bits must be discarded. As AES does not implement automatic data padding operation to **the last block**, the application must follow the recommendation given in [Section 22.4.4: AES procedure to perform a cipher operation on page 593](#) to manage messages the size of which is not a multiple of 128 bits.

**Note:** *Padding data are swapped in a similar way as normal data, according to the DATATYPE[1:0] field of the AES\_CR register (see [Section 22.4.13: AES data registers and data swapping](#) for details).*

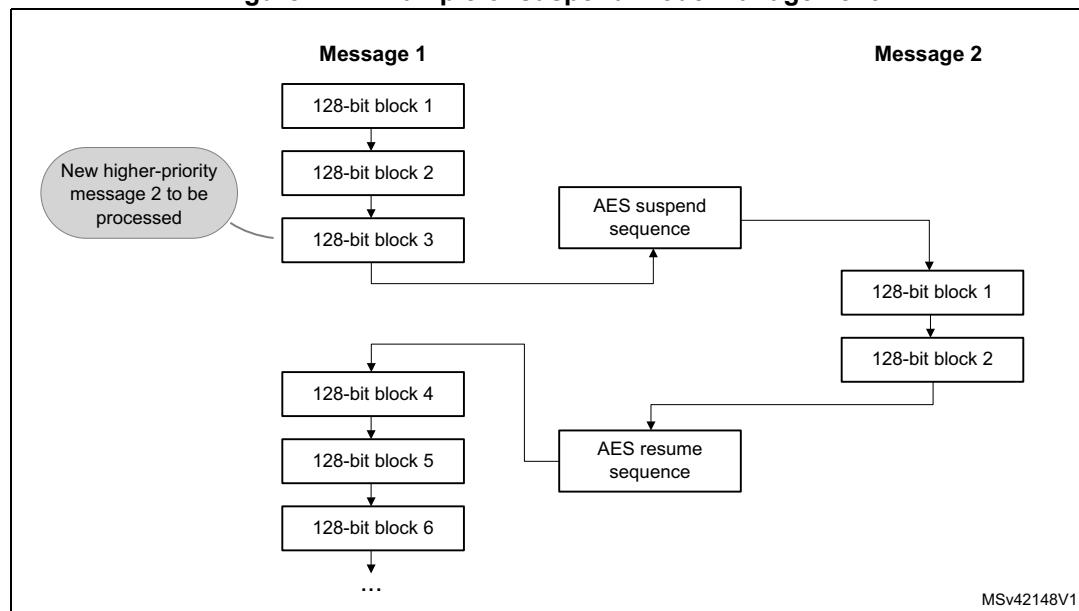
## 22.4.7 AES task suspend and resume

A message can be suspended if another message with a higher priority must be processed. When this highest priority message is sent, the suspended message can resume in both encryption or decryption mode.

Suspend/resume operations do not break the chaining operation and the message processing can resume as soon as AES is enabled again to receive the next data block.

[Figure 124](#) gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

**Figure 124. Example of suspend mode management**



A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

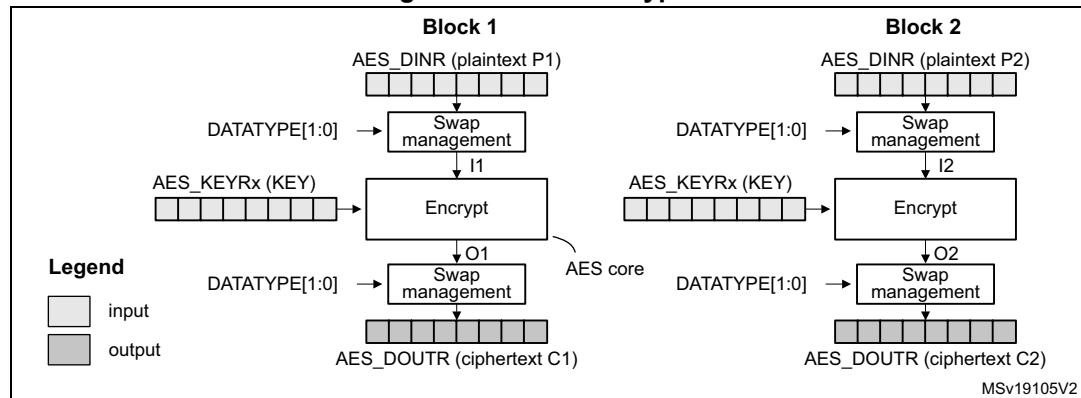
## 22.4.8 AES basic chaining modes (ECB, CBC)

### Overview

This section gives a brief explanation of the four basic operation modes provided by the AES core: ECB encryption, ECB decryption, CBC encryption and CBC decryption. For detailed information, refer to the FIPS publication 197 from November 26, 2001.

*Figure 125* illustrates the electronic codebook (ECB) encryption.

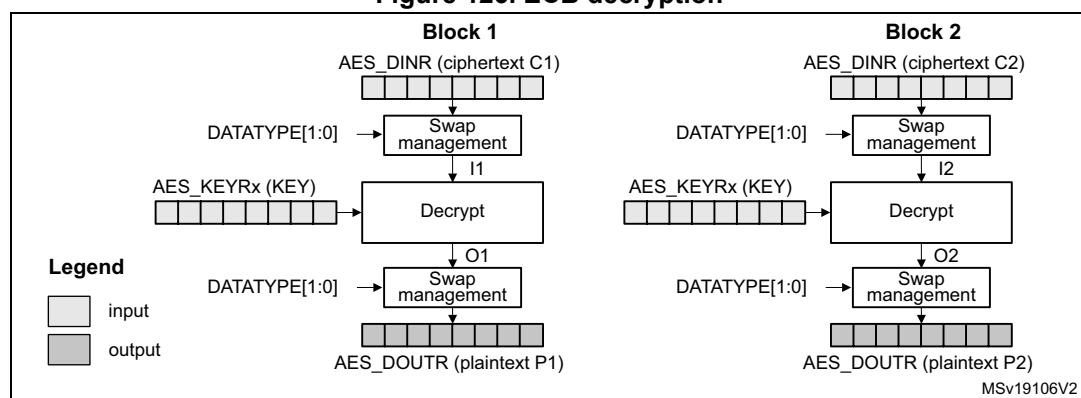
**Figure 125. ECB encryption**



In ECB encrypt mode, the 128-bit plaintext input data block Px in the AES\_DINR register first goes through bit/byte/half-word swapping. The swap result Ix is processed with the AES core set in encrypt mode, using a 128- or 256-bit key. The encryption result Ox goes through bit/byte/half-word swapping, then is stored in the AES\_DOUTR register as 128-bit ciphertext output data block Cx. The ECB encryption continues in this way until the last complete plaintext block is encrypted.

*Figure 126* illustrates the electronic codebook (ECB) decryption.

**Figure 126. ECB decryption**

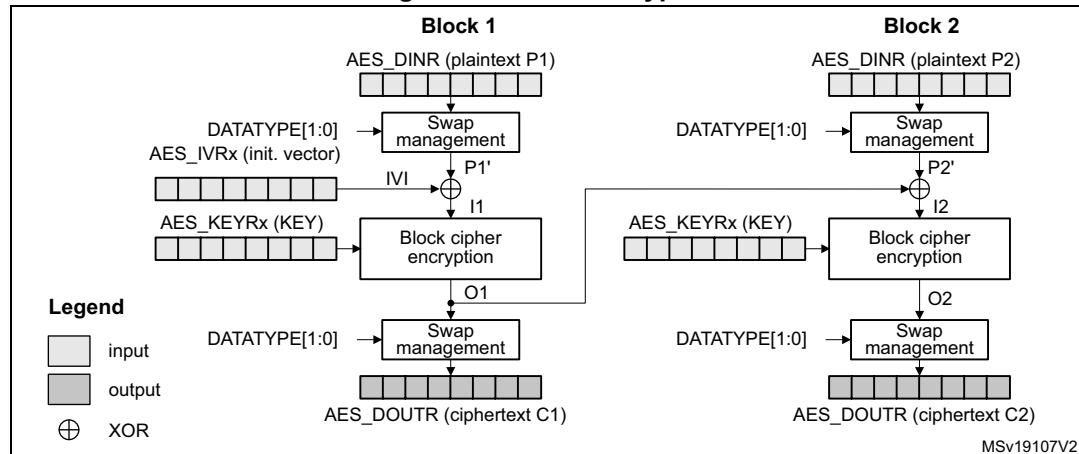


To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported by the AES core.

In ECB decrypt mode, the 128-bit ciphertext input data block C1 in the AES\_DINR register first goes through bit/byte/half-word swapping. The keying sequence is reversed compared to that of the ECB encryption. The swap result I1 is processed with the AES core set in decrypt mode, using the formerly prepared decryption key. The decryption result goes through bit/byte/half-word swapping, then is stored in the AES\_DOUTR register as 128-bit plaintext output data block P1. The ECB decryption continues in this way until the last complete ciphertext block is decrypted.

*Figure 127* illustrates the cipher block chaining (CBC) encryption.

**Figure 127. CBC encryption**

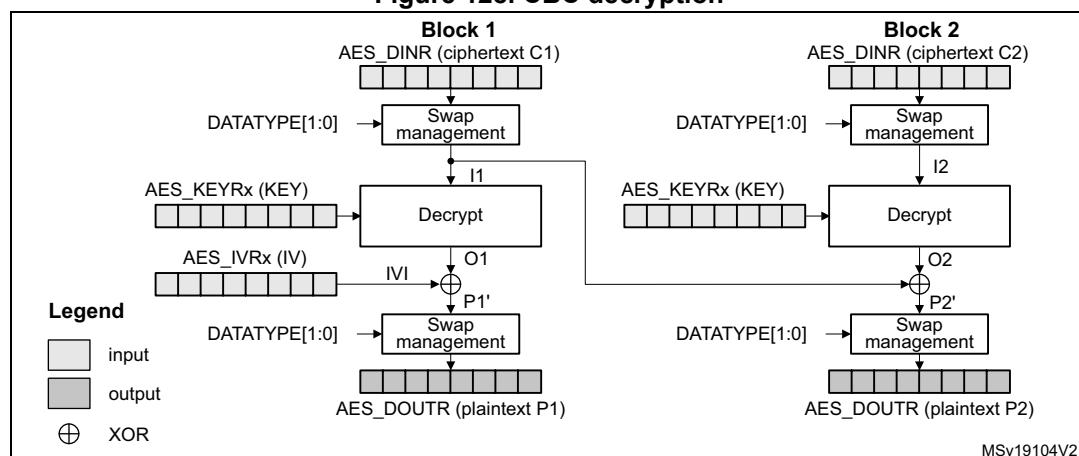


In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping (P1'), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the I1 input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block O1, after swapping operation, is used as ciphertext C1. The O1 data is then XOR-ed with the second-block plaintext data P2' to produce the I2 input data for the AES core to produce the second block of ciphertext data. The chaining of data blocks continues in this way until the last plaintext block in the message is encrypted.

If the message size is not a multiple of 128 bits, the final partial data block is encrypted in the way explained in [Section 22.4.6: AES ciphertext stealing and data padding](#).

*Figure 128* illustrates the cipher block chaining (CBC) decryption.

**Figure 128. CBC decryption**



In CBC decrypt mode, like in ECB decrypt mode, the secret key must be prepared to perform an AES decryption.

After the key preparation process, the decryption goes as follows: the first 128-bit ciphertext block (after the swap operation) is used directly as the AES core input block I1 for decrypt operation, using the 128-bit or 256-bit key. Its output O1 is XOR-ed with the 128-bit IV1 field (that must be identical to that used during encryption) to produce the first plaintext block P1.

The second ciphertext block is processed in the same way as the first block, except that the I1 data from the first block is used in place of the initialization vector.

The decryption continues in this way until the last complete ciphertext block is decrypted.

If the message size is not a multiple of 128 bits, the final partial data block is decrypted in the way explained in [Section 22.4.6: AES ciphertext stealing and data padding](#).

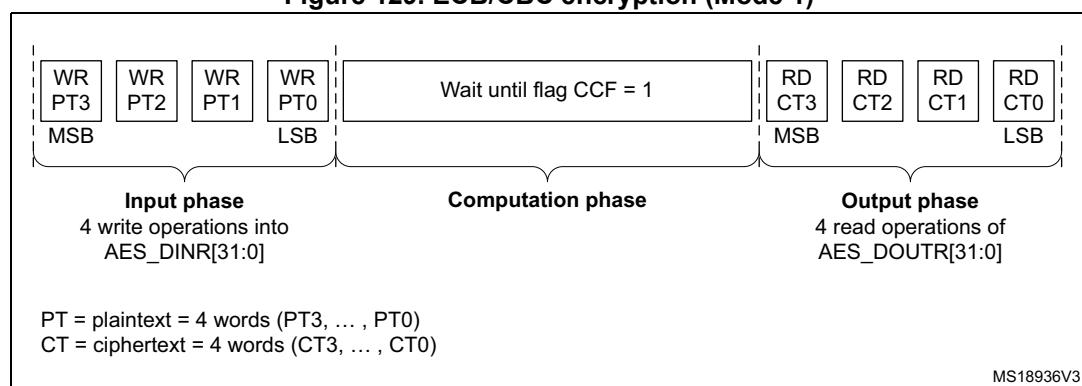
For more information on data swapping, refer to [Section 22.4.13: AES data registers and data swapping](#).

### ECB/CBC encryption sequence

The sequence of events to perform an ECB/CBC encryption (more detail in [Section 22.4.4](#)):

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select the Mode 1 by setting to 00 the MODE[1:0] bitfield of the AES\_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES\_CR register to 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield.
3. Select 128- or 256-bit key length through the KEYSIZE bit of the AES\_CR register.
4. Write the AES\_KEYRx registers (128 or 256 bits) with encryption key. Fill the AES\_IVRx registers with the initialization vector data if CBC mode has been selected.
5. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
6. Write the AES\_DINR register four times to input the plaintext (MSB first), as shown in [Figure 129](#).
7. Wait until the CCF flag is set in the AES\_SR register.
8. Read the AES\_DOUTR register four times to get the ciphertext (MSB first) as shown in [Figure 129](#). Then clear the CCF flag by setting the CCFC bit of the AES\_CR register.
9. Repeat steps 6-7-8 to process all the blocks with the same encryption key.

**Figure 129. ECB/CBC encryption (Mode 1)**

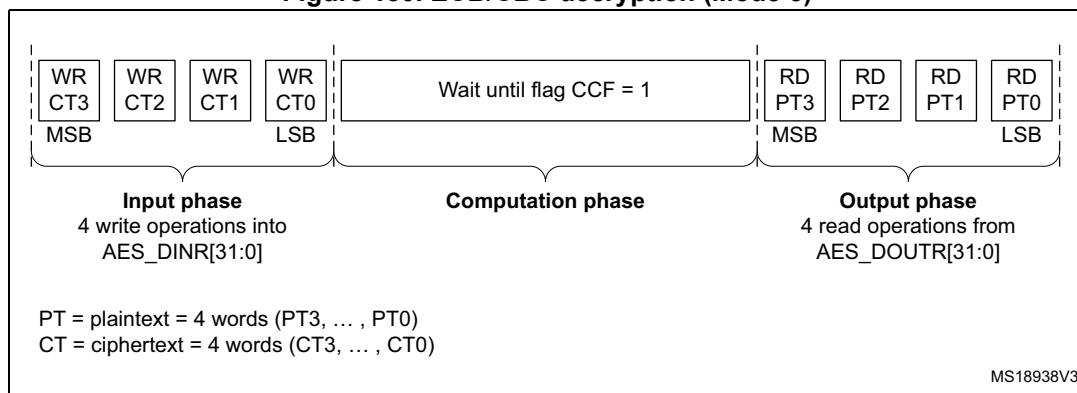


## ECB/CBC decryption sequence

The sequence of events to perform an AES ECB/CBC decryption is as follows (More detail in [Section 22.4.4](#)).

1. Follow the steps described in [Section 22.4.5: AES decryption round key preparation](#), in order to prepare the decryption key in AES core.
2. Select the Mode 3 by setting to 10 the MODE[1:0] bitfield of the AES\_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES\_CR register to 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield. KEYSIZE bitfield must be kept as-is.
3. Write the AES\_IVRx registers with the initialization vector (required in CBC mode only).
4. Enable AES by setting the EN bit of the AES\_CR register.
5. Write the AES\_DINR register four times to input the cipher text (MSB first), as shown in [Figure 130](#).
6. Wait until the CCF flag is set in the AES\_SR register.
7. Read the AES\_DOUTR register four times to get the plain text (MSB first), as shown in [Figure 130](#). Then clear the CCF flag by setting the CCFC bit of the AES\_CR register.
8. Repeat steps [5-6-7](#) to process all the blocks encrypted with the same key.

**Figure 130. ECB/CBC decryption (Mode 3)**



## Suspend/resume operations in ECB/CBC modes

**To suspend the processing of a message**, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES\_CR register.
2. If DMA is not used, read four times the AES\_DOUTR register to save the last processed block. If DMA is used, wait until the CCF flag is set in the AES\_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES\_CR register.
3. If DMA is not used, poll the CCF flag of the AES\_SR register until it becomes 1 (computation completed).
4. Clear the CCF flag by setting the CCFC bit of the AES\_CR register.
5. Save initialization vector registers (only required in CBC mode as AES\_IVRx registers are altered during the data processing).

6. Disable the AES peripheral by clearing the bit EN of the AES\_CR register.
7. Save the AES\_CR register and clear the key registers if they are not needed, to process the higher priority message.
8. If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, and so on).

**Note:** *In point 7, the derived key information stored in AES\_KEYRx registers can optionally be saved in memory if the interrupted process is a decryption. Otherwise those registers do not need to be saved as the original key value is known by the application*

**To resume the processing of a message**, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
3. Restore AES\_CR register (with correct KEYSIZE) then restore AES\_KEYRx registers. In case of decryption, derived key information can be written in AES\_KEYRx register instead of the original key value.
4. Prepare the decryption key as described in [Section 22.4.5: AES decryption round key preparation](#) (only required for ECB or CBC decryption). This step is not necessary if derived key information is loaded in AES\_KEYRx registers.
5. Restore AES\_IVRx registers using the saved configuration (only required in CBC mode).
6. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
7. If DMA is used, enable AES DMA transfers by setting the DMAINEN and DMAOUTEN bits of the AES\_CR register.

#### Alternative single ECB/CBC decryption using Mode 4

The sequence of events to perform a single round of ECB/CBC decryption using Mode 4 is:

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select the Mode 4 by setting to 11 the MODE[1:0] bitfield of the AES\_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES\_CR register to 0x0 or 0x1, respectively.
3. Select key length of 128 or 256 bits via KEYSIZE bitfield of the AES\_CR register.
4. Write the AES\_KEYRx registers with the encryption key. Write the AES\_IVRx registers if the CBC mode is selected.
5. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
6. Write the AES\_DINR register four times to input the cipher text (MSB first).
7. Wait until the CCF flag is set in the AES\_SR register.
8. Read the AES\_DOUTR register four times to get the plain text (MSB first). Then clear the CCF flag by setting the CCFC bit of the AES\_CR register.

**Note:** *When mode 4 is selected mode 3 cannot be used.*

*In mode 4, the AES\_KEYRx registers contain the encryption key during all phases of the processing. No derivation key is stored in these registers. It is stored internally in AES.*

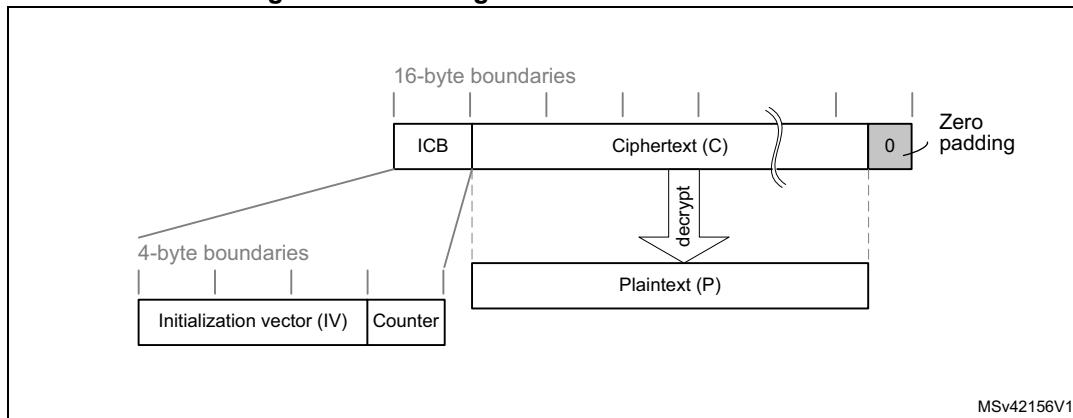
## 22.4.9 AES counter (CTR) mode

### Overview

The counter mode (CTR) uses AES as a key-stream generator. The generated keys are then XOR-ed with the plaintext to obtain the ciphertext.

CTR chaining is defined in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*. A typical message construction in CTR mode is given in [Figure 131](#).

**Figure 131. Message construction in CTR mode**



The structure of this message is:

- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector (IV)**: a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

### CTR encryption and decryption

[Figure 132](#) and [Figure 133](#) describe the CTR encryption and decryption process, respectively, as implemented in the AES peripheral. The CTR mode is selected by writing 010 to the CHMOD[2:0] bitfield of AES\_CR register.

Figure 132. CTR encryption

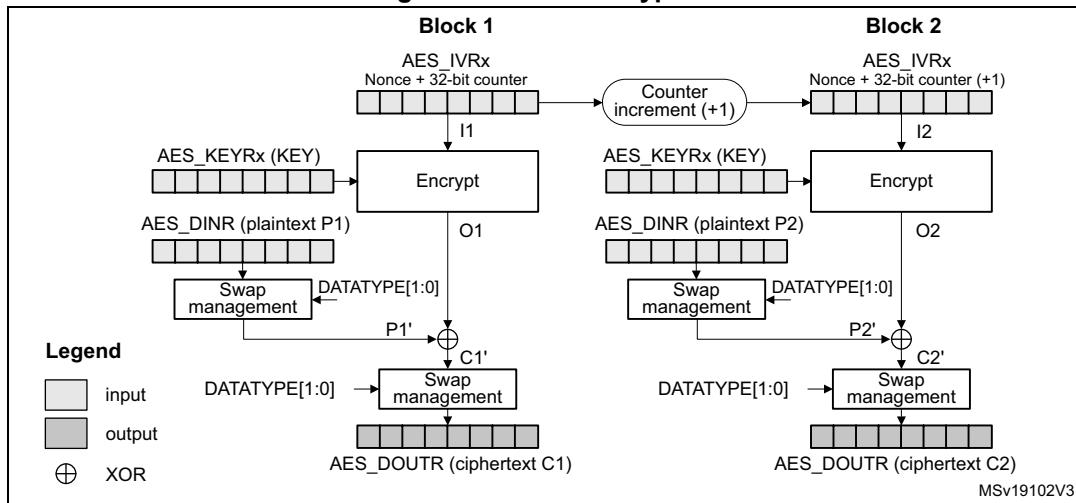
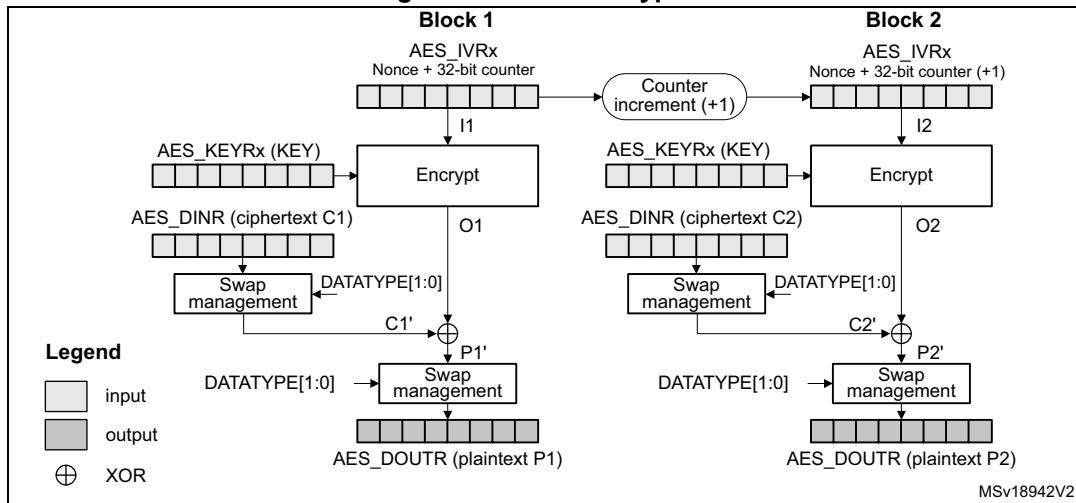


Figure 133. CTR decryption



In CTR mode, the cryptographic core output (also called keystream)  $O_x$  is XOR-ed with relevant input block ( $P_x'$  for encryption,  $C_x'$  for decryption), to produce the correct output block ( $C_x'$  for encryption,  $P_x'$  for decryption). Initialization vectors in AES must be initialized as shown in [Table 117](#).

Table 117. CTR mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

Unlike in CBC mode that uses the AES\_IVRx registers only once when processing the first data block, in CTR mode AES\_IVRx registers are used for processing each data block, and the AES peripheral increments the counter bits of the initialization vector (leaving the nonce bits unchanged).

CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that is then XOR-ed with the plaintext (CTR encryption) or ciphertext (CTR decryption) input. In CTR mode, the MODE[1:0] bitfield setting 01 (key derivation) is forbidden and all the other settings default to encryption mode.

The sequence of events to perform an encryption or a decryption in CTR chaining mode:

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select CTR chaining mode by setting to 010 the CHMOD[2:0] bitfield of the AES\_CR register. Set MODE[1:0] bitfield to any value other than 01.
3. Initialize the AES\_KEYRx registers, and load the AES\_IVRx registers as described in [Table 117](#).
4. Set the EN bit of the AES\_CR register, to start encrypting the current counter (EN is automatically reset when the calculation finishes).
5. If it is the last block, pad the data with zeros to have a complete block, if needed.
6. Append data in AES, and read the result. The three possible scenarios are described in [Section 22.4.4: AES procedure to perform a cipher operation](#).
7. Repeat the previous step till the second-last block is processed. For the last block, apply the two previous steps and discard the bits that are not part of the payload (if the size of the significant data in the last input block is less than 16 bytes).

### Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, and resume the message that was interrupted. Detailed CBC suspend/resume sequence is described in [Section 22.4.8: AES basic chaining modes \(ECB, CBC\)](#).

*Note:* *Like for CBC mode, the AES\_IVRx registers must be reloaded during the resume operation.*

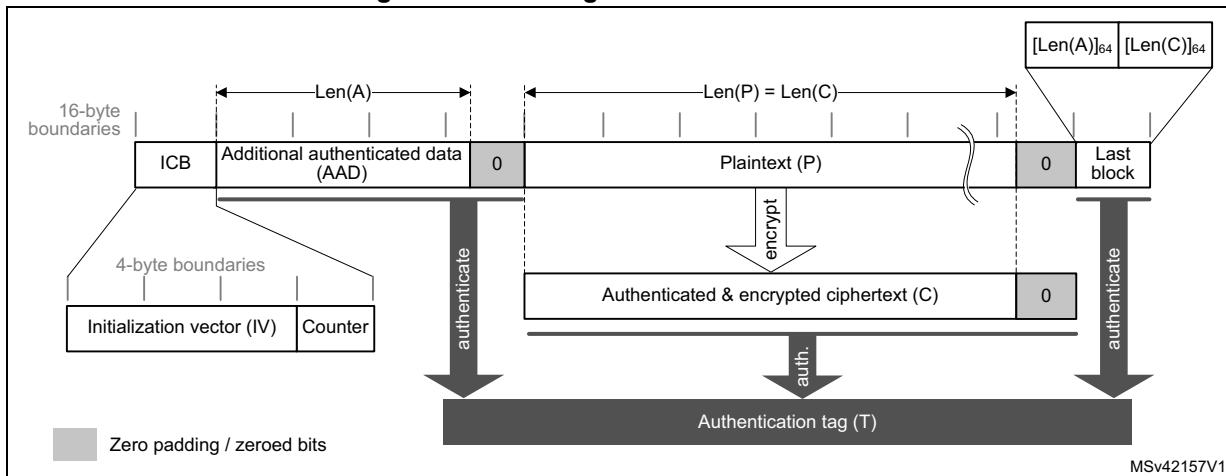
## 22.4.10 AES Galois/counter mode (GCM)

### Overview

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, GCM algorithm is based on AES counter mode. It uses a multiplier over a fixed finite field to generate the tag.

GCM chaining is defined in *NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*. A typical message construction in GCM mode is given in [Figure 134](#).

Figure 134. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV)**: a 96-bit value that must be unique for each encryption cycle with a given key. Note that the GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also known as additional authentication data) has a known length  $\text{Len}(A)$  that may be a non-multiple of 16 bytes, and must not exceed  $2^{64} - 1$  bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length  $\text{Len}(P)$  that may be non-multiple of 16 bytes, and cannot exceed  $2^{32} - 2$  128-bit blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in [Table 118](#).

The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

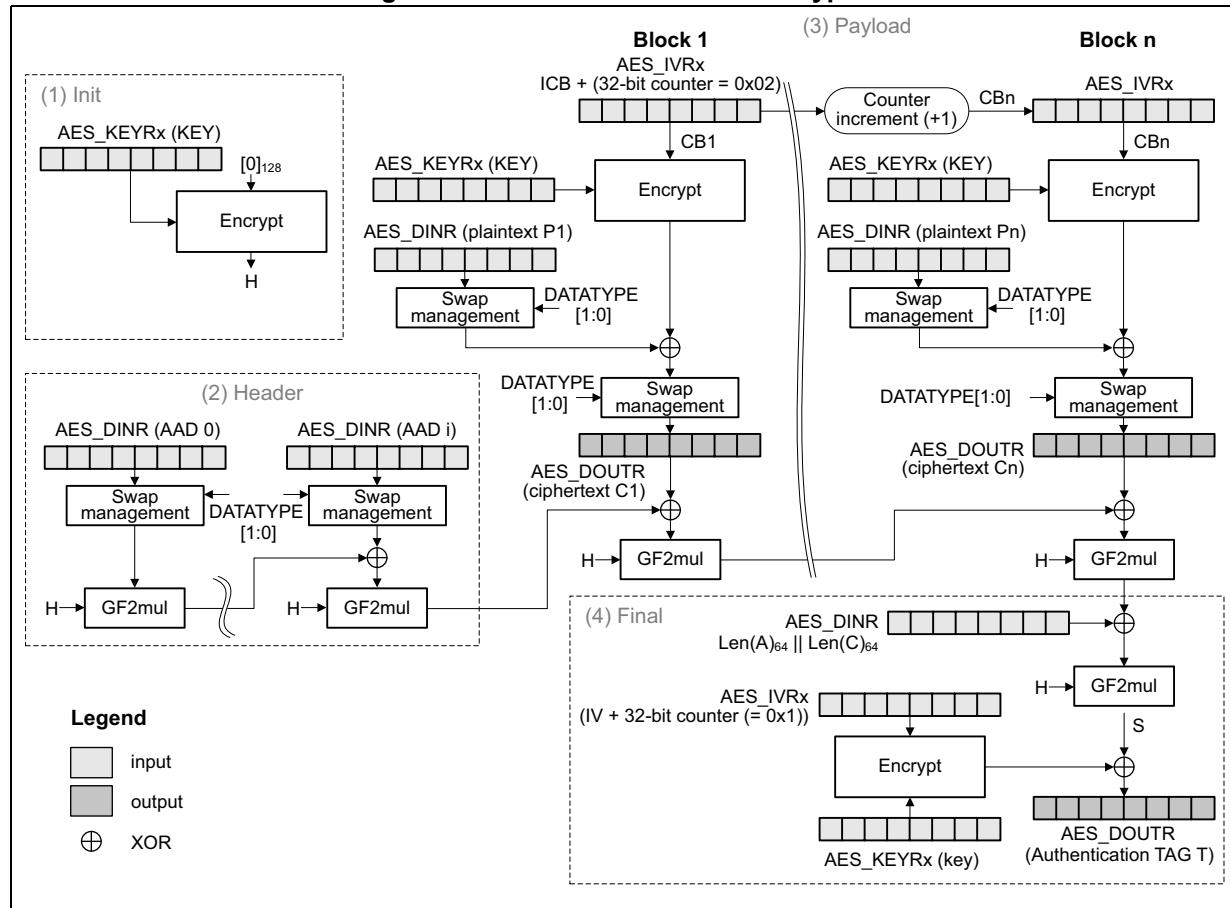
Table 118. GCM last block definition

Endianness	Bit[0] ----- Bit[31]	Bit[32]----- Bit[63]	Bit[64] ----- Bit[95]	Bit[96] ----- Bit[127]
Input data	0x0	AAD length[31:0]	0x0	Payload length[31:0]

## GCM processing

Figure 135 describes the GCM implementation in the AES peripheral. The GCM is selected by writing 011 to the CHMOD[2:0] bitfield of the AES\_CR register.

**Figure 135. GCM authenticated encryption**



The mechanism for the confidentiality of the plaintext in GCM mode is similar to that in the Counter mode, with a particular increment function (denoted 32-bit increment) that generates the sequence of input counter blocks.

AES\_IVRx registers keeping the **counter block** of data are used for processing each data block. The AES peripheral automatically increments the Counter[31:0] bitfield. The first counter block (CB1) is derived from the initial counter block ICB by the application software (see [Table 119](#)).

**Table 119. Initialization of AES\_IVRx registers in GCM mode**

Table 11: Initialization of AES_IVR registers in ROM mode			
AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

**Note:** *In this mode, the settings 01 and 11 of the MODE[1:0] bitfield are forbidden.*

The authentication mechanism in GCM mode is based on a hash function called **GF2mul** that performs multiplication by a fixed parameter, called hash subkey (H), within a binary Galois field.

A GCM message is processed through the following phases, further described in next subsections:

- **Init phase:** AES prepares the GCM hash subkey (H).
- **Header phase:** AES processes the additional authenticated data (AAD), with hash computation only.
- **Payload phase:** AES processes the plaintext (P) with hash computation, counter block encryption and data XOR-ing. It operates in a similar way for ciphertext (C).
- **Final phase:** AES generates the authenticated tag (T) using the last block of the message.

### GCM init phase

During this first step, the GCM hash subkey (H) is calculated and saved internally, to be used for processing all the blocks. The recommended sequence is:

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select GCM chaining mode, by setting to 011 the CHMOD[2:0] bitfield of the AES\_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES\_CR register.
4. Set the MODE[1:0] bitfield of the AES\_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES\_KEYRx registers with a key, and initialize AES\_IVRx registers with the information as defined in [Table 119](#).
6. Start the calculation of the hash key, by setting to 1 the EN bit of the AES\_CR register (EN is automatically reset when the calculation finishes).
7. Wait until the end of computation, indicated by the CCF flag of the AES\_SR transiting to 1. Alternatively, use the corresponding interrupt.
8. Clear the CCF flag of the AES\_SR register, by setting the CCFC bit of the AES\_CR register.

### GCM header phase

This phase coming after the GCM Init phase must be completed before the payload phase. The sequence to execute, identical for encryption and decryption, is:

1. Indicate the header phase, by setting to 01 the GCMPH[1:0] bitfield of the AES\_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
3. If it is the last block and the AAD size in the block is inferior to 128 bits, pad the remainder of the block with zeros. Then append the data block into AES in one of ways described in [Section 22.4.4: AES procedure to perform a cipher operation](#). No data is read during this phase.
4. Repeat the step 3 until the last additional authenticated data block is processed.

*Note:* *The header phase can be skipped if there is no AAD, that is, Len(A) = 0.*

### GCM payload phase

This phase, identical for encryption and decryption, is executed after the GCM header phase. During this phase, the encrypted/decrypted payload is stored in the AES\_DOUTR register. The sequence to execute is:

1. Indicate the payload phase, by setting to 10 the GCMPH[1:0] bitfield of the AES\_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES\_CR register.
3. If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is inferior to 128 bits, pad the remainder of the block with zeros.
4. Append the data block into AES in one of ways described in [Section 22.4.4: AES procedure to perform a cipher operation on page 593](#), and read the result.
5. Repeat the previous step till the second-last plaintext block is encrypted or till the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), execute the two previous steps. For the last block, discard the bits that are not part of the payload when the last block size is less than 16 bytes.

**Note:** *The payload phase can be skipped if there is no payload data, that is, Len(C) = 0 (see GMAC mode).*

### GCM final phase

In this last phase, the AES peripheral generates the GCM authentication tag and stores it in the AES\_DOUTR register. The sequence to execute is:

1. Indicate the final phase, by setting to 11 the GCMPH[1:0] bitfield of the AES\_CR register.
2. Compose the data of the block, by concatenating the AAD bit length and the payload bit length, as shown in [Table 118](#). Write the block into the AES\_DINR register.
3. Wait until the end of computation, indicated by the CCF flag of the AES\_SR transiting to 1.
4. Get the GCM authentication tag, by reading the AES\_DOUTR register four times.
5. Clear the CCF flag of the AES\_SR register, by setting the CCFC bit of the AES\_CR register.
6. Disable the AES peripheral, by clearing the bit EN of the AES\_CR register. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

**Note:** *In the final phase, data is written to AES\_DINR normally (no swapping), while swapping is applied to tag data read from AES\_DOUTR.*

*When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.*

### Suspend/resume operations in GCM mode

**To suspend the processing of a message**, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES\_CR register. If DMA is not used, make sure that the current computation is completed, which is indicated by the CCF flag of the AES\_SR register set to 1.
2. In the payload phase, if DMA is not used, read four times the AES\_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag is set in the AES\_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES\_CR register.
3. Clear the CCF flag of the AES\_SR register, by setting the CCFC bit of the AES\_CR register.
4. Save the AES\_SUSPxR registers in the memory, where x is from 0 to 7.
5. In the payload phase, save the AES\_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
6. Disable the AES peripheral, by clearing the EN bit of the AES\_CR register.
7. Save the current AES configuration in the memory, excluding the initialization vector registers AES\_IVRx. Key registers do not need to be saved as the original key value is known by the application.
8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

**To resume the processing of a message**, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.
2. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES\_SUSPxR registers, where x is from 0 to 7.
4. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES\_IVRx registers. In the header phase, write initial setting values back into the AES\_IVRx registers.
5. Restore the initial setting values in the AES\_CR and AES\_KEYRx registers.
6. Enable the AES peripheral by setting the EN bit of the AES\_CR register.

If DMA is used, enable AES DMA requests by setting the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES\_CR register.

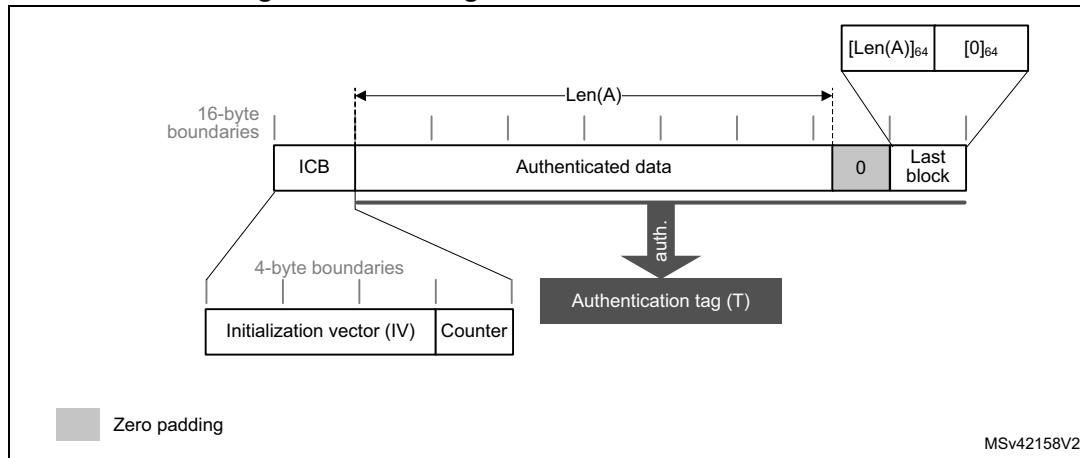
#### 22.4.11 AES Galois message authentication code (GMAC)

##### Overview

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code). It is based on GCM algorithm, as defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.

A typical message construction for GMAC is given in [Figure 136](#).

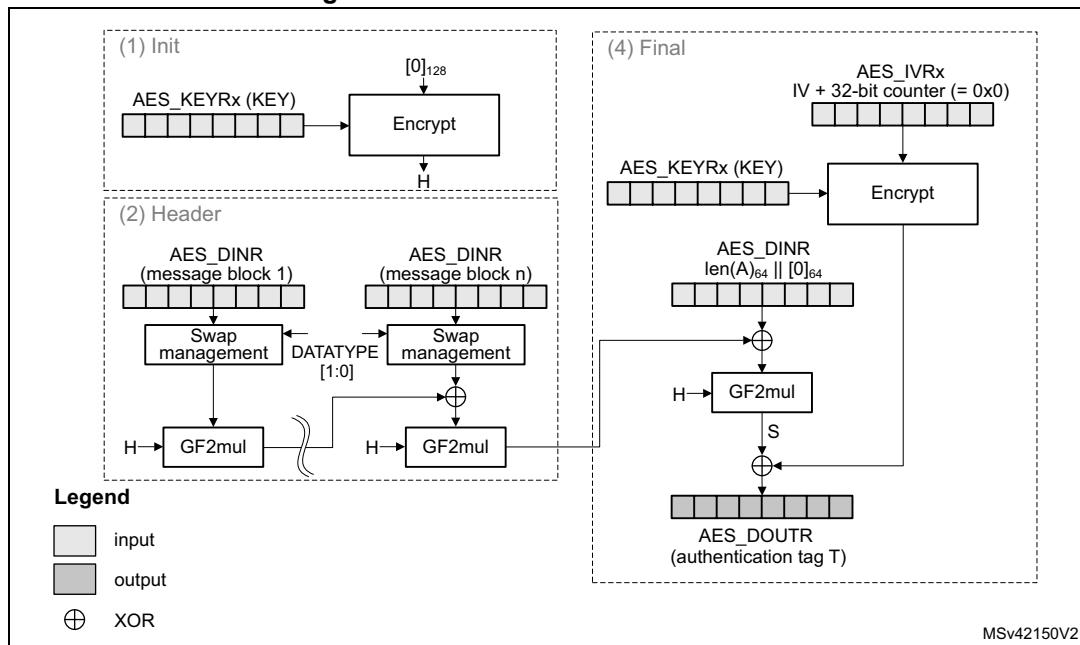
**Figure 136. Message construction in GMAC mode**



### AES GMAC processing

[Figure 137](#) describes the GMAC mode implementation in the AES peripheral. This mode is selected by writing 011 to the CHMOD[2:0] bitfield of the AES\_CR register.

**Figure 137. GMAC authentication mode**



The GMAC algorithm corresponds to the GCM algorithm applied on a message only containing a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

### Suspend/resume operations in GMAC

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

#### 22.4.12 AES counter with CBC-MAC (CCM)

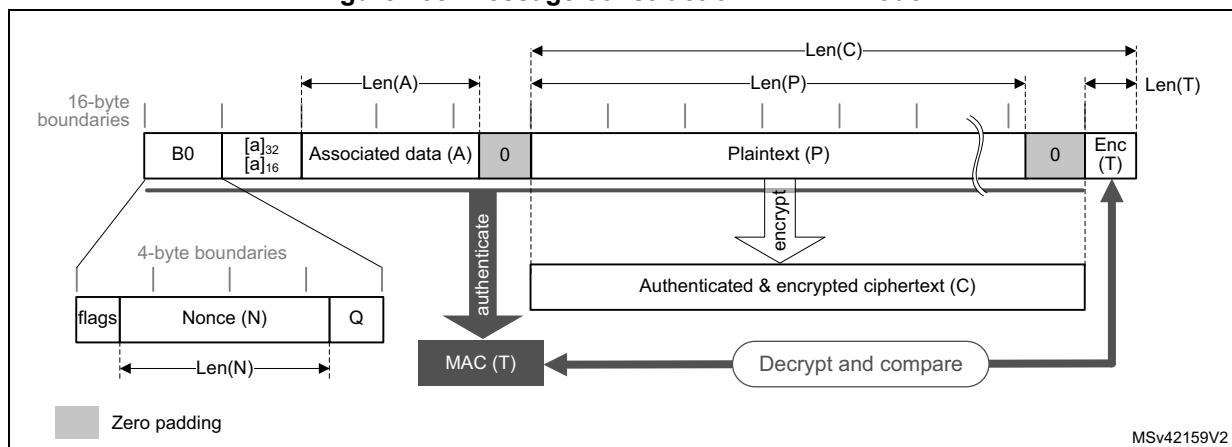
##### Overview

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES in counter mode. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

**Note:** *NIST does not approve this CBC-MAC as an authentication mode outside the context of the CCM specification.*

CCM chaining is specified in NIST Special Publication 800-38C, *Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*. A typical message construction for CCM is given in [Figure 138](#).

**Figure 138. Message construction in CCM mode**



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q**: a bit string representation of the octet length of P (Len(P))
  - **Nonce (N)**: a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags**: most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) <  $2^{8Q-4}$  bytes). The counter blocks range associated to **Q** is equal to  $2^{8Q-4}$ , that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the Associated Data (A). This part of the message is only authenticated, not encrypted. This section has a

known length  $\text{Len}(A)$  that can be a non-multiple of 16 bytes (see [Figure 138](#)). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:

- If  $0 < a < 2^{16} - 2^8$ , then it is encoded as  $[a]_{16}$ , that is, on two bytes.
- If  $2^{16} - 2^8 < a < 2^{32}$ , then it is encoded as  $0xff \parallel 0xfe \parallel [a]_{32}$ , that is, on six bytes.
- If  $2^{32} < a < 2^{64}$ , then it is encoded as  $0xff \parallel 0xff \parallel [a]_{64}$ , that is, on ten bytes.
- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length  $\text{Len}(P)$ . This length can be a non-multiple of 16 bytes (see [Figure 138](#)).
- **Encrypted MAC (T)** of length  $\text{Len}(T)$  appended to the ciphertext C of overall length  $\text{Len}(C)$ .

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

*Note:* *CCM chaining mode can also be used with associated data only (that is, no payload).*

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

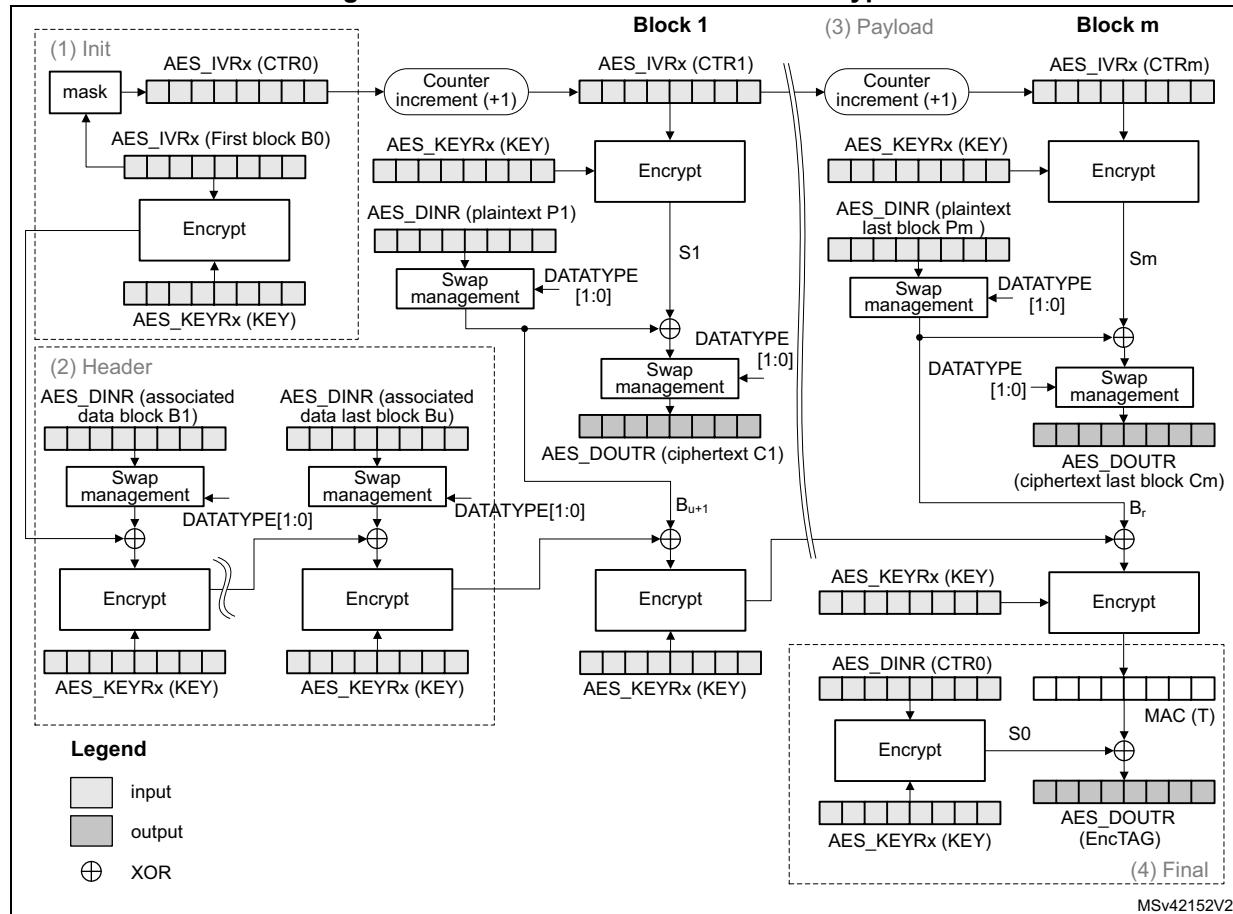
N: 10111213 141516 ( $\text{Len}(N) = 56$  bits or 7 bytes)  
A: 00010203 04050607 ( $\text{Len}(A) = 64$  bits or 8 bytes)  
P: 20212223 ( $\text{Len}(P) = 32$  bits or 4 bytes)  
T: 6084341B ( $\text{Len}(T) = 32$  bits or  $t = 4$ )  
B0: 4F101112 13141516 00000000 00000004  
B1: 00080001 02030405 06070000 00000000  
B2: 20212223 00000000 00000000 00000000  
CTR0: 0710111213 141516 00000000 00000000  
CTR1: 0710111213 141516 00000000 00000001

Generation of formatted input data blocks Bx (especially B0 and B1) must be managed by the application.

## CCM processing

[Figure 139](#) describes the CCM implementation within the AES peripheral (encryption example). This mode is selected by writing 100 into the CHMOD[2:0] bitfield of the AES\_CR register.

**Figure 139. CCM mode authenticated encryption**



The data input to the generation-encryption process are a valid nonce, a valid payload string, and a valid associated data string, all properly formatted. The CBC chaining mechanism is applied to the formatted plaintext data to generate a MAC, with a known length. Counter mode encryption that requires a sufficiently long sequence of counter blocks as input, is applied to the payload string and separately to the MAC. The resulting ciphertext C is the output of the generation-encryption process on plaintext P.

AES\_IVRx registers are used for processing each data block, AES automatically incrementing the CTR counter with a bit length defined by the first block B0. [Table 120](#) shows how the application must load the B0 data.

**Note:** The AES peripheral in CCM mode supports counters up to 64 bits, as specified by NIST.

**Table 120. Initialization of AES\_IVRx registers in CCM mode**

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
B0[127:96]	B0[95:64]	B0[63:32]	B0[31:0]

**Note:** *In this mode, the settings 01 and 11 of the MODE[1:0] bitfield are forbidden.*

A CCM message is processed through the following phases, further described in next subsections:

- **Init phase:** AES processes the first block and prepares the first counter block.
- **Header phase:** AES processes associated data (A), with tag computation only.
- **Payload phase:** IP processes plaintext (P), with tag computation, counter block encryption, and data XOR-ing. It works in a similar way for ciphertext (C).
- **Final phase:** AES generates the message authentication code (MAC).

### CCM Init phase

In this phase, the first block B0 of the CCM message is written into the AES\_IVRx register. The AES\_DOUTR register does not contain any output data. The recommended sequence is:

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select CCM chaining mode, by setting to 100 the CHMOD[2:0] bitfield of the AES\_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES\_CR register.
4. Set the MODE[1:0] bitfield of the AES\_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES\_KEYRx registers with a key, and initialize AES\_IVRx registers with B0 data as described in [Table 120](#).
6. Start the calculation of the counter, by setting to 1 the EN bit of the AES\_CR register (EN is automatically reset when the calculation finishes).
7. Wait until the end of computation, indicated by the CCF flag of the AES\_SR transiting to 1. Alternatively, use the corresponding interrupt.
8. Clear the CCF flag in the AES\_SR register, by setting to 1 the CCFC bit of the AES\_CR register.

### CCM header phase

This phase coming after the GCM Init phase must be completed before the payload phase. During this phase, the AES\_DOUTR register does not contain any output data.

The sequence to execute, identical for encryption and decryption, is:

1. Indicate the header phase, by setting to 01 the GCMPH[1:0] bitfield of the AES\_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
3. If it is the last block and the AAD size in the block is inferior to 128 bits, pad the remainder of the block with zeros. Then append the data block into AES in one of ways described in [Section 22.4.4: AES procedure to perform a cipher operation](#). No data is read during this phase.
4. Repeat the step 3 until the last additional authenticated data block is processed.

**Note:**

*The header phase can be skipped if there is no associated data, that is, Len(A) = 0.*

*The first block of the associated data (B1) must be formatted by software, with the associated data length.*

### CCM payload phase (encryption or decryption)

This phase, identical for encryption and decryption, is executed after the CCM header phase. During this phase, the encrypted/decrypted payload is stored in the AES\_DOUTR register. The sequence to execute is:

1. Indicate the payload phase, by setting to 10 the GCM[1:0] bitfield of the AES\_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES\_CR register.
3. If it is the last data block to encrypt and the plaintext size in the block is inferior to 128 bits, pad the remainder of the block with zeros.
4. Append the data block into AES in one of ways described in [Section 22.4.4: AES procedure to perform a cipher operation on page 593](#), and read the result.
5. Repeat the previous step till the second-last plaintext block is encrypted or till the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), apply the two previous steps. For the last block, discard the data that is not part of the payload when the last block size is less than 16 bytes.

**Note:** *The payload phase can be skipped if there is no payload data, that is, Len(P) = 0 or Len(C) = Len(T).*

*Remove LSB<sub>Len(T)</sub>(C) encrypted tag information when decrypting ciphertext C.*

### CCM final phase

In this last phase, the AES peripheral generates the GCM authentication tag and stores it in the AES\_DOUTR register. The sequence to execute is:

1. Indicate the final phase, by setting to 11 the GCM[1:0] bitfield of the AES\_CR register.
2. Wait until the end-of-computation flag CCF of the AES\_SR register is set.
3. Read four times the AES\_DOUTR register: the output corresponds to the CCM authentication tag.
4. Clear the CCF flag of the AES\_SR register by setting the CCFC bit of the AES\_CR register.
5. Disable the AES peripheral, by clearing the EN bit of the AES\_CR register.
6. For authenticated decryption, compare the generated encrypted tag with the encrypted tag padded in the ciphertext.

**Note:** *In this final phase, swapping is applied to tag data read from AES\_DOUTR register.*

*When transiting from the header phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.*

*Application must mask the authentication tag output with tag length to obtain a valid tag.*

### Suspend/resume operations in CCM mode

**To suspend the processing of a message** in header or payload phase, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES\_CR register. If DMA is not used, make sure that the current computation is completed, which is indicated by the CCF flag of the AES\_SR register set to 1.
2. In the payload phase, if DMA is not used, read four times the AES\_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag is set in the

- AES\_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES\_CR register.
3. Clear the CCF flag of the AES\_SR register, by setting to 1 the CCFC bit of the AES\_CR register.
  4. Save the AES\_SUSPxR registers (where x is from 0 to 7) in the memory.
  5. Save the AES\_IVRx registers as, during the data processing, they changed from their initial values.
  6. Disable the AES peripheral, by clearing the EN bit of the AES\_CR register.
  7. Save the current AES configuration in the memory, excluding the initialization vector registers AES\_IVRx. Key registers do not need to be saved as the original key value is known by the application.
  8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

**To resume the processing of a message**, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.
2. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES\_SUSPxR registers (where x is from 0 to 7).
4. Write the initialization vector register values, previously saved in the memory, back into their corresponding AES\_IVRx registers.
5. Restore the initial setting values in the AES\_CR and AES\_KEYRx registers.
6. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
7. If DMA is used, enable AES DMA requests by setting to 1 the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES\_CR register.

## 22.4.13 AES data registers and data swapping

### Data input and output

A 128-bit data block is entered into the AES peripheral with four successive 32-bit word writes into the AES\_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 128-bit data block is retrieved from the AES peripheral with four successive 32-bit word reads from the AES\_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The 32-bit data word for AES\_DINR register or from AES\_DOUTR register is organized in big endian order, that is:

- the most significant byte of a word to write into AES\_DINR must be put on the lowest address out of the four adjacent memory locations keeping the word to write, or
- the most significant byte of a word read from AES\_DOUTR goes to the lowest address out of the four adjacent memory locations receiving the word

For using DMA for input data block write into AES, the four words of the input block must be stored in the memory consecutively and in big-endian order, that is, the most significant word on the lowest address. See [Section 22.4.16: AES DMA interface](#).

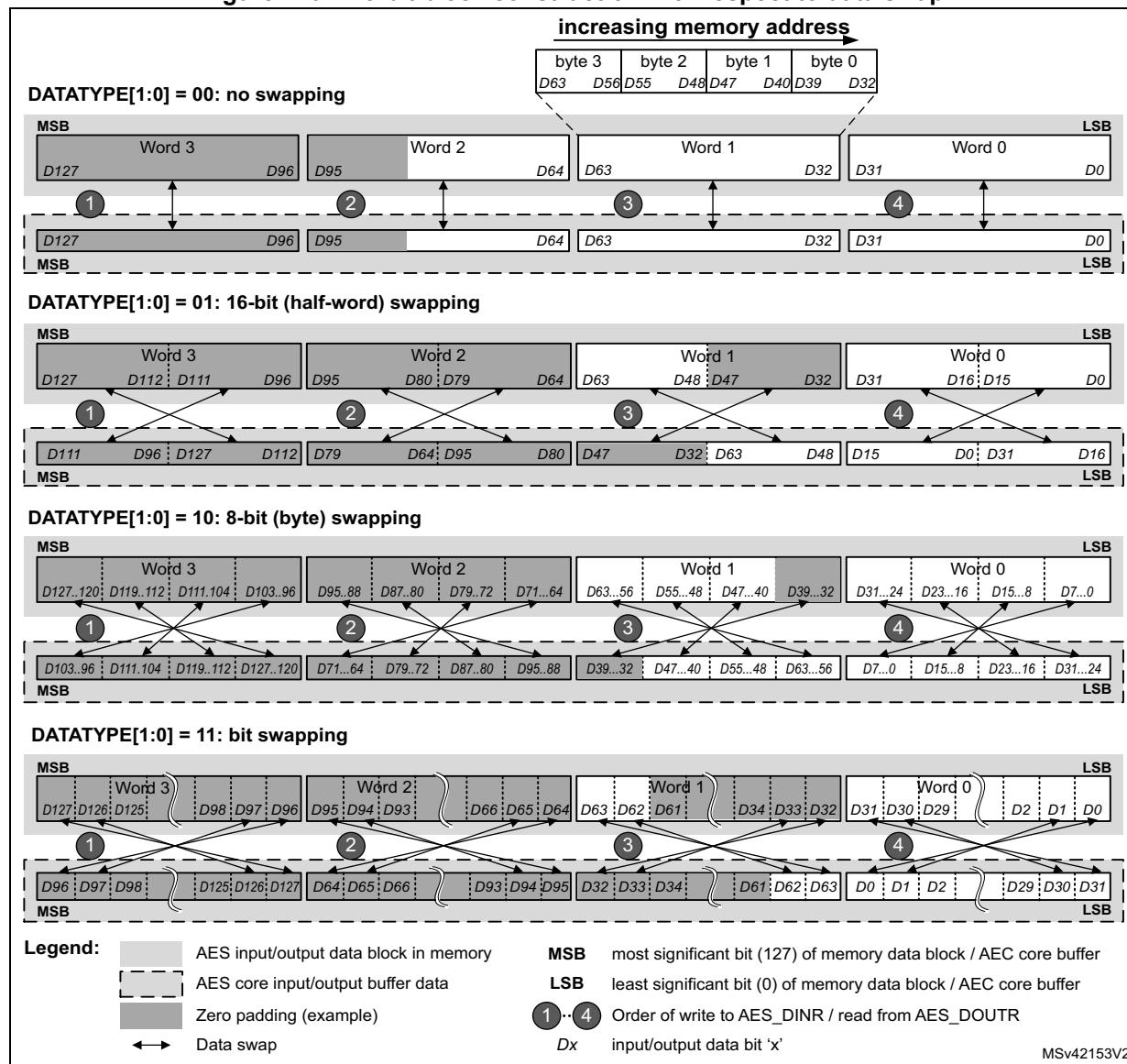
## Data swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES\_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES\_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through the DATATYPE[1:0] bitfield of the AES\_CR register. The selection applies both to the input and the output of the AES core.

For different data swap types, [Figure 140](#) shows the construction of AES processing core input buffer data P127 to P0, from the input data entered through the AES\_DINR register, or the construction of the output data available through the AES\_DOUTR register, from the AES processing core output buffer data P127 to P0.

**Figure 140. 128-bit block construction with respect to data swap**



**Note:** The data in AES key registers (AES\_KEYRx) and initialization registers (AES\_IVRx) are not sensitive to the swap mode selection.

### Data padding

[Figure 140](#) also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 48 message bits, with DATATYPE[1:0] = 01
- 56 message bits, with DATATYPE[1:0] = 10
- 34 message bits, with DATATYPE[1:0] = 11

#### 22.4.14 AES key registers

The AES\_KEYRx registers store the encryption or decryption key bitfield KEY[127:0] or KEY[255:0]. The data to write to or to read from each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address.

The key is spread over eight registers as shown in [Table 121](#).

**Table 121. Key endianness in AES\_KEYRx registers (128- or 256-bit key length)**

AES_KEYR7 [31:0]	AES_KEYR6 [31:0]	AES_KEYR5 [31:0]	AES_KEYR4 [31:0]	AES_KEYR3 [31:0]	AES_KEYR2 [31:0]	AES_KEYR1 [31:0]	AES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]

The key for encryption or decryption may be written into these registers when the AES peripheral is disabled, by clearing the EN bit of the AES\_CR register.

The key registers are not affected by the data swapping controlled by DATATYPE[1:0] bitfield of the AES\_CR register.

#### 22.4.15 AES initialization vector registers

The four AES\_IVRx registers keep the initialization vector input bitfield IVI[127:0]. The data to write to or to read from each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address. The registers are also ordered from lowest address (AES\_IVR0) to highest address (AES\_IVR3).

The signification of data in the bitfield depends on the chaining mode selected. When used, the bitfield is updated upon each computation cycle of the AES core.

Write operations to the AES\_IVRx registers when the AES peripheral is enabled have no effect to the register contents. For modifying the contents of the AES\_IVRx registers, the EN bit of the AES\_CR register must first be cleared.

Reading the AES\_IVRx registers returns the latest counter value (useful for managing suspend mode).

The AES\_IVRx registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield of the AES\_CR register.

### 22.4.16 AES DMA interface

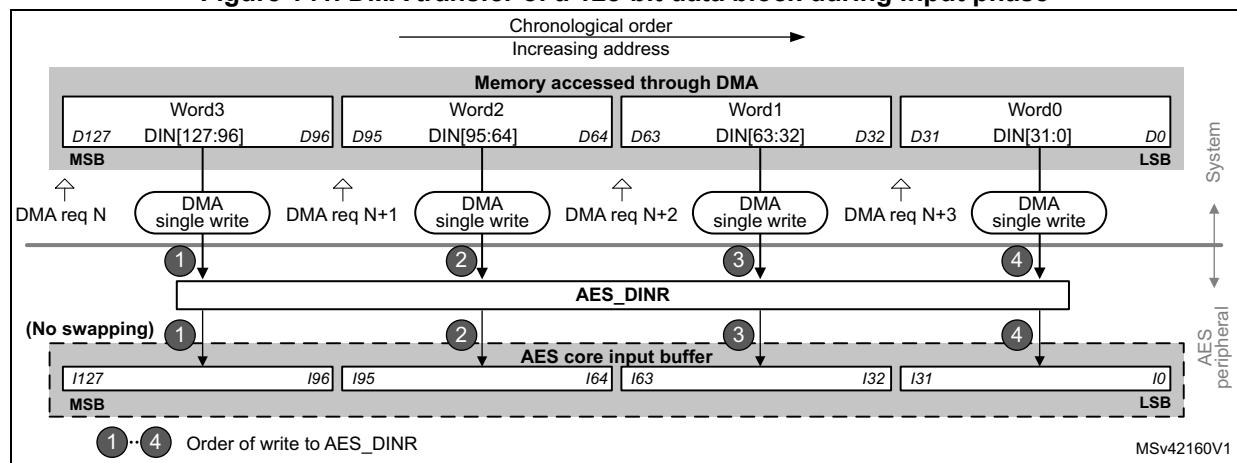
The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the AES\_CR register.

#### Data input using DMA

Setting the DMAINEN bit of the AES\_CR register enables DMA writing into AES. The AES peripheral then initiates a DMA request during the input phase each time it requires to write a 128-bit block (quadruple word) to the AES\_DINR register, as shown in [Figure 141](#).

**Note:** *According to the algorithm and the mode selected, special padding / ciphertext stealing might be required. For example, in case of AES GCM encryption or AES CCM decryption, a DMA transfer must not include the last block. For details, refer to [Section 22.4.4: AES procedure to perform a cipher operation](#).*

**Figure 141. DMA transfer of a 128-bit data block during input phase**

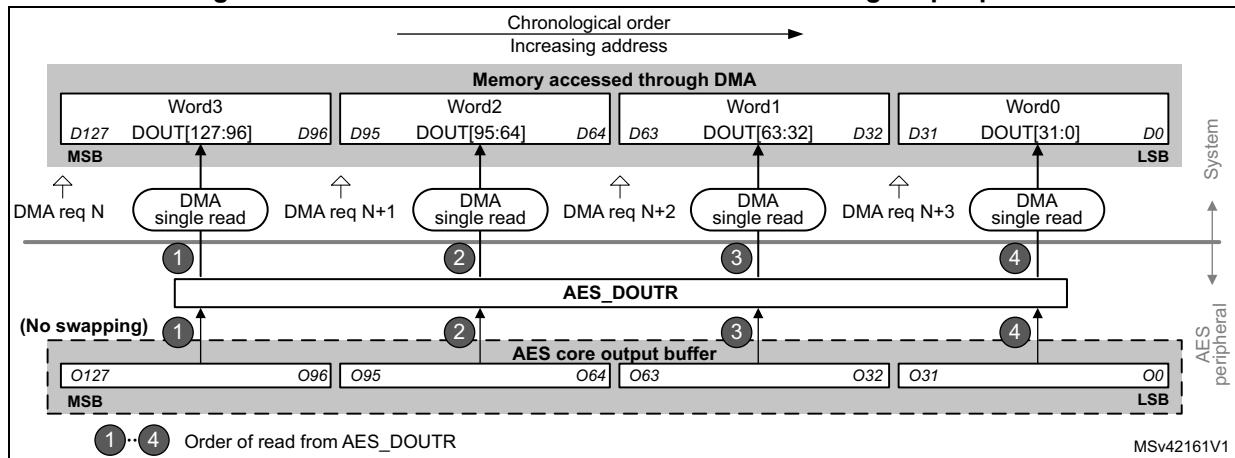


#### Data output using DMA

Setting the DMAOUTEN bit of the AES\_CR register enables DMA reading from AES. The AES peripheral then initiates a DMA request during the Output phase each time it requires to read a 128-bit block (quadruple word) to the AES\_DINR register, as shown in [Figure 142](#).

**Note:** *According to the message size, extra bytes might need to be discarded by application in the last block.*

Figure 142. DMA transfer of a 128-bit data block during output phase



### DMA operation in different operating modes

DMA operations are usable when Mode 1 (encryption) or Mode 3 (decryption) are selected via the MODE[1:0] bitfield of the register AES\_CR. As in Mode 2 (key derivation) the AES\_KEYRx registers must be written by software, enabling the DMA transfer through the DMAINEN and DMAOUTEN bits of the AES\_CR register have no effect in that mode.

DMA single requests are generated by AES until it is disabled. So, after the data output phase at the end of processing of a 128-bit data block, AES switches automatically to a new data input phase for the next data block, if any.

When the data transferring between AES and memory is managed by DMA, the CCF flag has no use because the reading of the AES\_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to data transferring managed by software. See [Section 22.4.4: AES procedure to perform a cipher operation](#), subsection [Data append](#), for details.

## 22.4.17 AES error management

AES configuration can be changed at any moment by clearing the EN bit of the AES\_CR register.

### Read error flag (RDERR)

Unexpected read attempt of the AES\_DOUTR register sets the RDERR flag of the AES\_SR register, and returns zero.

RDERR is triggered during the computation phase or during the input phase.

*Note:* AES is not disabled upon a RDERR error detection and continues processing.

An interrupt is generated if the ERRIE bit of the AES\_CR register is set. For more details, refer to [Section 22.5: AES interrupts](#).

The RDERR flag is cleared by setting the ERRIE bit of the AES\_CR register.

### Write error flag (WDERR)

Unexpected write attempt of the AES\_DINR register sets the WRERR flag of the AES\_SR register, and has no effect on the AES\_DINR register. The WRERR is triggered during the computation phase or during the output phase.

**Note:** *AES is not disabled after a WRERR error detection and continues processing.*

An interrupt is generated if the ERRIE bit of the AES\_CR register is set. For more details, refer to [Section 22.5: AES interrupts](#).

The WRERR flag is cleared by setting the ERRC bit of the AES\_CR register.

## 22.5 AES interrupts

Individual maskable interrupt sources generated by the AES peripheral signal the following events:

- computation completed
- read error
- write error

The individual sources are combined into the common interrupt signal aes\_it that connects to NVIC (nested vectored interrupt controller). Each can individually be enabled/disabled, by setting/clearing the corresponding enable bit of the AES\_CR register, and cleared by setting the corresponding bit of the AES\_CR register.

The status of each can be read from the AES\_SR register.

[Table 122](#) gives a summary of the interrupt sources, their event flags and enable bits.

**Table 122. AES interrupt requests**

Interrupt acronym	AES interrupt event	Event flag	Enable bit	Interrupt clear method
AES	computation completed flag	CCF	CCFIE	set CCFC <sup>(1)</sup>
	read error flag	RDERR	ERRIE	set ERRC <sup>(1)</sup>
	write error flag	WRERR		

1. Bit of the AES\_CR register.

## 22.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

**Table 123. Processing latency for ECB, CBC and CTR**

Key size	Mode of operation	Algorithm	Clock cycles
128-bit	Mode 1: Encryption	ECB, CBC, CTR	51
	Mode 2: Key derivation	-	59
	Mode 3: Decryption	ECB, CBC, CTR	51
	Mode 4: Key derivation then decryption	ECB, CBC	106

**Table 123. Processing latency for ECB, CBC and CTR (continued)**

Key size	Mode of operation	Algorithm	Clock cycles
256-bit	Mode 1: Encryption	ECB, CBC, CTR	75
	Mode 2: Key derivation	-	82
	Mode 3: Decryption	ECB, CBC, CTR	75
	Mode 4: Key derivation then decryption	ECB, CBC	145

**Table 124. Processing latency for GCM and CCM (in clock cycles)**

Key size	Mode of operation	Algorithm	Init Phase	Header phase <sup>(1)</sup>	Payload phase <sup>(1)</sup>	Tag phase <sup>(1)</sup>
128-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	64	35	51	59
		CCM	63	55	114	58
256-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	88	35	75	75
		CCM	87	79	162	82

1. Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).

## 22.7 AES registers

### 22.7.1 AES control register (AES\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
								rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **NPBLB[3:0]**: Number of padding bytes in last block

The bitfield sets the number of padding bytes in last block of payload:

0000: All bytes are valid (no padding)

0001: Padding for one least-significant byte of last block

...

1111: Padding for 15 least-significant bytes of last block

Bit 19 Reserved, must be kept at reset value.

**Bit 18 KEYSIZE:** Key size selection

This bitfield defines the length of the key used in the AES cryptographic core, in bits:

- 0: 128
- 1: 256

Attempts to write the bit are ignored when the EN bit of the AES\_CR register is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

**Bits 14:13 GCM/CCM PHASE[1:0]:** GCM or CCM phase selection

This bitfield selects the phase of GCM, GMAC or CCM algorithm:

- 00: Init phase
- 01: Header phase
- 10: Payload phase
- 11: Final phase

The bitfield has no effect if other than GCM, GMAC or CCM algorithms are selected (through the ALGOMODE bitfield).

**Bit 12 DMAOUTEN:** DMA output enable

This bit enables/disables data transferring with DMA, in the output phase:

- 0: Disable
- 1: Enable

When the bit is set, DMA requests are automatically generated by AES during the output data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).

Use of DMA with Mode 4 (single decryption) is not recommended.

**Bit 11 DMAINEN:** DMA input enable

This bit enables/disables data transferring with DMA, in the input phase:

- 0: Disable
- 1: Enable

When the bit is set, DMA requests are automatically generated by AES during the input data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).

Use of DMA with Mode 4 (single decryption) is not recommended.

**Bit 10 ERRIE:** Error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when RDERR and/or WRERR is set:

- 0: Disable (mask)
- 1: Enable

**Bit 9 CCFIE:** CCF interrupt enable

This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set:

- 0: Disable (mask)
- 1: Enable

**Bit 8 ERRC:** Error flag clear

Upon written to 1, this bit clears the RDERR and WRERR error flags in the AES\_SR register:

- 0: No effect
- 1: Clear RDERR and WRERR flags

Reading the flag always returns zero.

Bit 7 **CCFC**: Computation complete flag clear

Upon written to 1, this bit clears the computation complete flag (CCF) in the AES\_SR register:

0: No effect

1: Clear CCF

Reading the flag always returns zero.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode selection

This bitfield selects the AES chaining mode:

000: Electronic codebook (ECB)

001: Cipher-block chaining (CBC)

010: Counter mode (CTR)

011: Galois counter mode (GCM) and Galois message authentication code (GMAC)

100: Counter with CBC-MAC (CCM)

others: Reserved

Attempts to write the bitfield are ignored when the EN bit of the AES\_CR register is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: AES operating mode

This bitfield selects the AES operating mode:

00: Mode 1: encryption

01: Mode 2: key derivation (or key preparation for ECB/CBC decryption)

10: Mode 3: decryption

11: Mode 4: key derivation then single decryption

Attempts to write the bitfield are ignored when the EN bit of the AES\_CR register is set before the write access and it is not cleared by that write access. Any attempt to selecting Mode 4 while either ECB or CBC chaining mode is not selected, defaults to effective selection of Mode 3. It is not possible to select a Mode 3 following a Mode 4.

Bits 2:1 **DATATYPE[1:0]**: Data type selection

This bitfield defines the format of data written in the AES\_DINR register or read from the AES\_DOUTR register, through selecting the mode of data swapping:

00: None

01: Half-word (16-bit)

10: Byte (8-bit)

11: Bit

For more details, refer to [Section 22.4.13: AES data registers and data swapping](#).

Attempts to write the bitfield are ignored when the EN bit of the AES\_CR register is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: AES enable

This bit enables/disables the AES peripheral:

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the AES peripheral.

This bit is automatically cleared by hardware upon the completion of the key preparation (Mode 2) and upon the completion of GCM/GMAC/CCM initial phase.

## 22.7.2 AES status register (AES\_SR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

### Bit 3 **BUSY**: Busy

This flag indicates whether AES is idle or busy during GCM payload **encryption** phase:

0: Idle

1: Busy

When the flag indicates “idle”, the current GCM encryption processing may be suspended to process a higher-priority message. In other chaining modes, or in GCM phases other than payload encryption, the flag must be ignored for the suspend process.

### Bit 2 **WRERR**: Write error

This flag indicates the detection of an unexpected write operation to the AES\_DINR register (during computation or data output phase):

0: Not detected

1: Detected

The flag is set by hardware. It is cleared by software upon setting the ERRC bit of the AES\_CR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES\_CR register.

The flag setting has no impact on the AES operation. Unexpected write is ignored.

### Bit 1 **RDERR**: Read error flag

This flag indicates the detection of an unexpected read operation from the AES\_DOUTR register (during computation or data input phase):

0: Not detected

1: Detected

The flag is set by hardware. It is cleared by software upon setting the ERRC bit of the AES\_CR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES\_CR register.

The flag setting has no impact on the AES operation. Unexpected read returns zero.

### Bit 0 **CCF**: Computation completed flag

This flag indicates whether the computation is completed:

0: Not completed

1: Completed

The flag is set by hardware upon the completion of the computation. It is cleared by software, upon setting the CCFC bit of the AES\_CR register.

Upon the flag setting, an interrupt is generated if enabled through the CCFIE bit of the AES\_CR register.

The flag is significant only when the DMAOUTEN bit is 0. It may stay high when DMA\_EN is 1.

### 22.7.3 AES data input register (AES\_DINR)

Address offset: 0x08

Reset value: 0x0000 0000

Only 32-bit access type is supported.

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

Bits 31:0 **DIN[31:0]**: Input data word

A four-fold sequential write to this bitfield during the input phase results in writing a complete 128-bit block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 128-bit input buffer.

The data signification of the input data block depends on the AES operating mode:

- **Mode 1** (encryption): plaintext
- **Mode 2** (key derivation): the bitfield is not used (AES\_KEYRx registers used for input)
- **Mode 3** (decryption) and **Mode 4** (key derivation then single decryption): ciphertext

The data swap operation is described in [Section 22.4.13: AES data registers and data swapping on page 617](#).

### 22.7.4 AES data output register (AES\_DOUTR)

Address offset: 0x0C

Reset value: 0x0000 0000

Only 32-bit read access type is supported.

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

Bits 31:0 **DOUT[31:0]**: Output data word

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF set), virtually reads a complete 128-bit block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

The data signification of the output data block depends on the AES operating mode:

- **Mode 1** (encryption): ciphertext
- **Mode 2** (key derivation): the bitfield is not used (AES\_KEYRx registers used for output)
- **Mode 3** (decryption) and **Mode 4** (key derivation then single decryption): plaintext

The data swap operation is described in [Section 22.4.13: AES data registers and data swapping on page 617](#).

## 22.7.5 AES key register 0 (AES\_KEYR0)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

This bitfield contains the bits [31:0] of the AES encryption or decryption key, depending on the operating mode:

- In **Mode 1** (encryption), **Mode 2** (key derivation) and **Mode 4** (key derivation then single decryption): the value to write into the bitfield is the encryption key.
- In **Mode 3** (decryption): the value to write into the bitfield is the encryption key to be derived before being used for decryption. After writing the encryption key into the bitfield, its reading before enabling AES returns the same value. Its reading after enabling AES and after the CCF flag is set returns the decryption key derived from the encryption key.

*Note: In mode 4 (key derivation then single decryption) the bitfield always contains the encryption key.*

The AES\_KEYRx registers may be written only when KEYSIZE value is correct and when the AES peripheral is disabled (EN bit of the AES\_CR register cleared). Note that, if, the key is directly loaded to AES\_KEYRx registers (hence writes to key register is ignored and KEIF is set).

Refer to [Section 22.4.14: AES key registers on page 619](#) for more details.

## 22.7.6 AES key register 1 (AES\_KEYR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
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
KEY[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

## 22.7.7 AES key register 2 (AES\_KEYR2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
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
KEY[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

## 22.7.8 AES key register 3 (AES\_KEYR3)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
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
KEY[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

## 22.7.9 AES initialization vector register 0 (AES\_IVR0)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

Refer to [Section 22.4.15: AES initialization vector registers on page 619](#) for description of the IVI[127:0] bitfield.

The initialization vector is only used in chaining modes other than ECB.

The AES\_IVRx registers may be written only when the AES peripheral is disabled

## 22.7.10 AES initialization vector register 1 (AES\_IVR1)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the AES\_IVR0 register for description of the IVI[128:0] bitfield.

## 22.7.11 AES initialization vector register 2 (AES\_IVR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the AES\_IVR0 register for description of the IVI[128:0] bitfield.

### 22.7.12 AES initialization vector register 3 (AES\_IVR3)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]

Refer to the AES\_IVR0 register for description of the IVI[128:0] bitfield.

### 22.7.13 AES key register 4 (AES\_KEYR4)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
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
KEY[143:128]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]

Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

### 22.7.14 AES key register 5 (AES\_KEYR5)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
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
KEY[175:160]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]

Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

### 22.7.15 AES key register 6 (AES\_KEYR6)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
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
KEY[207:192]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

### 22.7.16 AES key register 7 (AES\_KEYR7)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
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
KEY[239:224]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

**Note:** The key registers from 4 to 7 are used only when the key length of 256 bits is selected. They have no effect when the key length of 128 bits is selected (only key registers 0 to 3 are used in that case).

### 22.7.17 AES suspend registers (AES\_SUSPxR)

Address offset: 0x040 + x \* 0x4, (x = 0 to 7)

Reset value: 0x0000 0000

These registers contain the complete internal register states of the AES processor when the AES processing of the current task is suspended to process a higher-priority task.

Upon suspend, the software reads and saves the AES\_SUSPxR register contents (where x is from 0 to 7) into memory, before using the AES processor for the higher-priority task.

Upon completion, the software restores the saved contents back into the corresponding suspend registers, before resuming the original task.

**Note:** These registers are used only when GCM, GMAC, or CCM chaining mode is selected.

These registers can be read only when AES is enabled. Reading these registers while AES is disabled returns 0x0000 0000.

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

Bits 31:0 **SUSP[31:0]**: AES suspend

Upon suspend operation, this bitfield of the corresponding AES\_SUSPxR register takes the value of one of internal AES registers.

## 22.7.18 AES register map

Table 125. AES register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x000	AES_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]			Res.	Res.	Res.	Res.	
										0	0	0	0	0	0	0	
0x004	AES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x008	AES_DINR	DIN[31:0]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	AES_DOUTR	DOUT[31:0]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	AES_KEYR0	KEY[31:0]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	AES_KEYR1	KEY[63:32]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	AES_KEYR2	KEY[95:64]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	AES_KEYR3	KEY[127:96]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	AES_IVR0	IVI[31:0]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	AES_IVR1	IVI[63:32]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	AES_IVR2	IVI[95:64]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	AES_IVR3	IVI[127:96]															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 125. AES register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	AES_KEYR4																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x034	AES_KEYR5																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x038	AES_KEYR6																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x03C	AES_KEYR7																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x040	AES_SUSP0R																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x044	AES_SUSP1R																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x048	AES_SUSP2R																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x04C	AES_SUSP3R																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x050	AES_SUSP4R																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x054	AES_SUSP5R																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x058	AES_SUSP6R																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x05C	AES_SUSP7R																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x060-0x3FF	Reserved	Res.																															

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 23 Public key accelerator (PKA)

### 23.1 Introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over  $GF(p)$  (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

All needed computations are performed within the accelerator, so no further hardware/software elaboration is needed to process the inputs or the outputs.

### 23.2 PKA main features

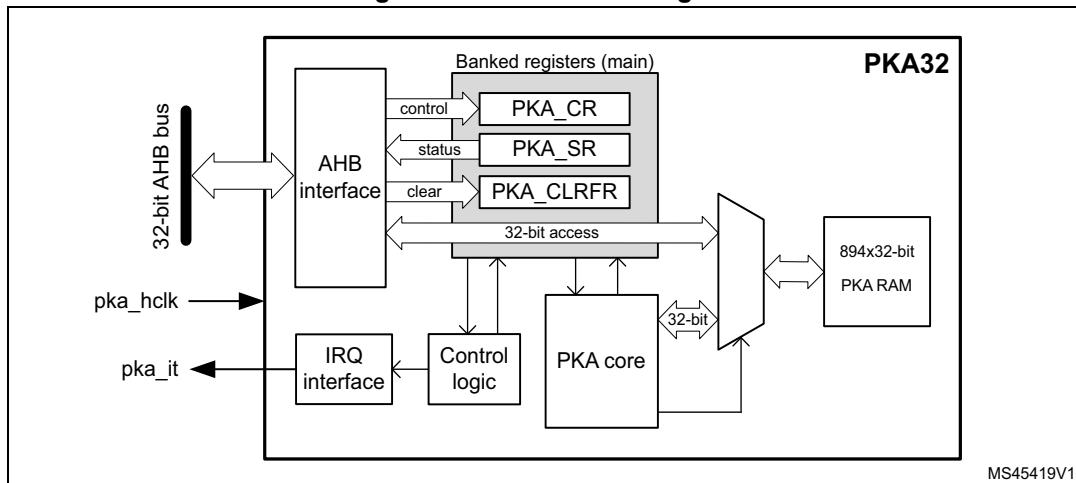
- Acceleration of RSA, DH and ECC over  $GF(p)$  operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA Chinese Remainder Theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check
  - ECDSA signature generation and verification
- Capability to handle operands up to 3136 bits for RSA/DH and 640 bits for ECC.
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication.
- Built-in Montgomery domain inward and outward transformations.
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise, for writes, an AHB bus error is generated, and write accesses are ignored).

### 23.3 PKA functional description

#### 23.3.1 PKA block diagram

*Figure 143* shows the block diagram of the public key accelerator PKA.

Figure 143. PKA block diagram



### 23.3.2 PKA internal signals

Table 126 lists internal signals available at the IP level, not necessarily available on product bonding pads.

Table 126. Internal input/output signals

Signal name	Signal type	Description
pka_hclk	Digital input	AHB bus clock
pka_it	Digital output	Public key accelerator IP global interrupt request

### 23.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. The RAM receives this clock directly, the core is clocked at half the frequency.

### 23.3.4 PKA public key acceleration

#### Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 3136 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation  $y^2 = x^3 + ax + b \pmod{p}$ . More information is found in [Section 23.5.1: Supported elliptic curves](#).

**Note:** *Binary curves, Edwards curves and Curve25519 are not supported by the PKA.*

A memory of 3576 bytes (894 words of 32 bits) called PKA RAM is used for providing initial data to the PKA, and for holding the results after computation is completed. Access is done through the PKA AHB interface.

## PKA operating modes

The list of operations the PKA can perform is detailed in [Table 127](#) and [Table 128](#), respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Each of these operating modes has an associated code that has to be written to the MODE field in the PKA\_CR register.

**Table 127. PKA integer arithmetic functions list**

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation $R2 \bmod n$	<a href="#">Section 23.4.2</a>
0x0E	001110	Modular addition $(A+B) \bmod n$	<a href="#">Section 23.4.3</a>
0x0F	001111	Modular subtraction $(A-B) \bmod n$	<a href="#">Section 23.4.4</a>
0x10	010000	Montgomery multiplication $(AxB) \bmod n$	<a href="#">Section 23.4.5</a>
0x00	000000	Modular exponentiation $A^e \bmod n$	<a href="#">Section 23.4.6</a>
0x02	000010	Modular exponentiation $A^e \bmod n$ (fast mode)	
0x08	001000	Modular inversion $A^{-1} \bmod n$	<a href="#">Section 23.4.7</a>
0x0D	001101	Modular reduction $A \bmod n$	<a href="#">Section 23.4.8</a>
0x09	001001	Arithmetic addition $A+B$	<a href="#">Section 23.4.9</a>
0x0A	001010	Arithmetic subtraction $A-B$	<a href="#">Section 23.4.10</a>
0x0B	001011	Arithmetic multiplication $AxB$	<a href="#">Section 23.4.11</a>
0x0C	001100	Arithmetic comparison $(A=B, A>B, A<B)$	<a href="#">Section 23.4.12</a>
0x07	000111	RSA CRT exponentiation	<a href="#">Section 23.4.13</a>

**Table 128. PKA prime field (Fp) elliptic curve functions list**

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x28	101000	Point on elliptic curve Fp check	<a href="#">Section 23.4.14</a>
0x20	100000	ECC scalar multiplication $kP$	<a href="#">Section 23.4.15</a>
0x22	100010	ECC scalar multiplication $kP$ (fast mode)	
0x24	100100	ECDSA sign	<a href="#">Section 23.4.16</a>
0x26	100110	ECDSA verification	<a href="#">Section 23.4.17</a>

## Montgomery space and fast mode operations

For efficiency reason the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming the application can decide to use a faster mode of operation, during which the precomputed Montgomery parameter is

supplied before starting the operation. Performance improvement is detailed in [Section 23.5.2: Computation times](#).

The operations using fast mode are modular exponentiation and scalar multiplication.

### 23.3.5 Typical applications for PKA

#### Introduction

The PKA can be used to accelerate a number of public key cryptographic functions. In particular:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

Specifications of the above functions are given in following publications:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

The principles of the main functions are described in this section, for a more detailed description refer to the above cited documents.

#### RSA key pair

For following RSA operations a public key and a private key information are defined as below:

- Alice transmits her public key  $(n, e)$  to Bob. Numbers  $n$  and  $e$  are very large positive integers.
- Alice keeps secret her private key  $d$ , also a very large positive integer. Alternatively this private key can also be represented by a quintuple  $(p, q, dp, dq, qInv)$ .

For more information on above representations refer to the RSA specification.

#### RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message  $M$  using Alice's public key  $(n, e)$  must go through the following steps:

1. Compute the encoded message  $EM = \text{ENCODE}(M)$ , where ENCODE is an encoding method.
2. Turn  $EM$  into an integer  $m$ , with  $0 \leq m < n$  and  $(m, n)$  being co-primes.
3. Compute ciphertext  $c = m^e \bmod n$ .
4. Convert the integer  $c$  into a string ciphertext  $C$ .

Alice, to decrypt ciphertext  $c$  using her private key, follows the steps indicated below:

1. Convert the ciphertext  $C$  to an integer ciphertext representative  $c$ .
2. Recover plaintext  $m = c^d \bmod n = (m^e)^d \bmod n$ . If the private key is the quintuple  $(p, q, dp, dq, qInv)$ , then plaintext  $m$  is obtained by performing the operations:
  - a)  $m_1 = c^{dp} \bmod p$
  - b)  $m_2 = c^{dq} \bmod q$
  - c)  $h = qInv (m_1 - m_2) \bmod p$
  - d)  $m = m_2 + h q$
3. Convert the integer message representative  $m$  to an encoded message  $EM$ .
4. Recover message  $M = \text{DECODE}(EM)$ , where  $\text{DECODE}$  is a decoding method.

Above operations can be accelerated by PKA using [Modular exponentiation](#)  $A^e \bmod n$  if the private key is  $d$ , or [RSA CRT exponentiation](#) if the private key is the quintuple  $(p, q, dp, dq, qInv)$ .

*Note:* The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

### Elliptic curve selection

For following ECC operations curve parameters are defined as below:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in [Section 23.5.1: Supported elliptic curves](#).
- $G$  is the chosen elliptic curve base point (also known as generator), with a large prime order  $n$  (i.e.  $n \times G = \text{identity element } O$ ).

### ECDSA message signature generation

ECDSA (Elliptic Curve Digital Signature Algorithm) signature generation function principle is the following: Alice, to sign a message  $m$  using her private key integer  $d_A$ , follows the steps below.

1. Calculate  $e = \text{HASH}(m)$ , where  $\text{HASH}$  is a cryptographic hash function.
2. Let  $z$  be the  $L_n$  leftmost bits of  $e$ , where  $L_n$  is the bit length of the group order  $n$ .
3. Select a cryptographically secure random integer  $k$  where  $0 < k < n$ .
4. Calculate the curve point  $(x_1, y_1) = k \times G$ .
5. Calculate  $r = x_1 \bmod n$ . If  $r = 0$  go back to step 3.
6. Calculate  $s = k^{-1} (z + rd_A) \bmod n$ . If  $s = 0$  go back to step 3.
7. The signature is the pair  $(r, s)$ .

Steps 4 to 7 are accelerated by PKA using:

- [ECDSA sign](#) or
- All of the operations below:
  - [ECC Fp scalar multiplication](#)  $k \times P$
  - [Modular reduction](#)  $A \bmod n$
  - [Modular inversion](#)  $A^{-1} \bmod n$
  - [Modular addition](#) and [Modular and Montgomery multiplication](#)

### ECDSA signature verification

ECDSA (elliptic curve digital signature algorithm) signature verification function principle is the following: Bob, to authenticate Alice's signature, must have a copy of her public key curve point  $Q_A$ .

Bob can verify that  $Q_A$  is a valid curve point going through the following steps:

1. check that  $Q_A$  is not equal to the identity element  $O$
2. check that  $Q_A$  is on the agreed curve
3. check that  $n \times Q_A = O$ .

Then Bob follows the procedure detailed below:

1. verify that  $r$  and  $s$  are integer in  $[1, n-1]$
2. calculate  $e = \text{HASH}(m)$ , where HASH is the agreed cryptographic hash function
3. let  $z$  be the  $L_n$  leftmost bits of  $e$
4. calculate  $w = s^{-1} \bmod n$
5. calculate  $u_1 = zw \bmod n$  and  $u_2 = rw \bmod n$
6. calculate the curve point  $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$
7. the signature is valid if  $r = x_1 \pmod n$ , it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using [ECDSA verification](#).

### 23.3.6 PKA procedure to perform an operation

#### Enabling/disabling PKA

Setting the EN bit to 1 in PKA\_CR register enables the PKA peripheral. When EN = 0, the PKA peripheral is kept under reset, with PKA memory still accessible by the application through the AHB interface.

Clearing EN bit to 0 while a calculation is in progress causes the operation to be aborted. In this case, the content of the PKA memory is not guaranteed.

#### Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in [Section 23.4](#).

#### Executing a PKA operation

Each of the supported PKA operation is executed using the following procedure:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE field of PKA\_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA\_CR register.
3. Wait until the PROCENDF bit in the PKA\_SR register is set to "1", indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA\_CLRFR.

**Note:** When PKA is busy (BUSY = 1) any access by the application to PKA RAM is ignored, and the flag RAMERRF is set in PKA\_SR.

### Using precomputed Montgomery parameters (PKA fast mode)

As explained in [Section 23.3.4](#), when computing many operations with the same modulus it can be beneficial for the application to compute only once the corresponding Montgomery parameter (see, for example, [Section 23.4.5](#)). This is known as “fast mode”.

To manage Fast Mode usage the recommended procedure is described below:

1. Load in PKA RAM the modulus size and value information. Such information is compiled in [Section 23.5.1](#).
2. Program in PKA\_CR register the PKA in [Montgomery parameter computation](#) mode (MODE="0x1") then assert the START bit.
3. Wait until the PROCENDF bit in the PKA\_SR register is set to “1”, then read back from PKA memory the corresponding Montgomery parameter, and then clear PROCENDF bit by setting PROCENDFC bit in PKA\_CLRFR.
4. Proceed with the required PKA operation, loading on top of regular input data the Montgomery information R2 mod m. All addresses are indicated in [Section 23.4](#).

### 23.3.7 PKA error management

When PKA is used some errors can occur:

- The access to PKA RAM falls outside the expected range. In this case the Address Error flag (ADDRERRF) is set in the PKA\_SR register.
- An AHB access to the PKA RAM occurred while the PKA core was using it. In this case the RAM Error Flag (RAMERRF) is set in the PKA\_SR register, reads to PKA RAM return zero, while writes are ignored.

For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA\_CR register (see [Section 23.6](#) for details).

ADDRERRF and RAMERRF errors are cleared by setting the corresponding bit in PKA\_CLRFR.

The PKA can be re-initialized at any moment by resetting the EN bit in the PKA\_CR register.

## 23.4 PKA operating modes

### 23.4.1 Introduction

The various operations supported by PKA are described in the following subsections, clarifying the associated format of the input data and of the results, both stored in the PKA RAM.

The following information applies to all PKA operations.

- PKA core processes 32-bit words
- Supported operand “Size” are:
  - ROS (RSA operand size): data size is  $(rsa\_size/32+1)$  words, with  $rsa\_size$  equal to the chosen modulus length. For example, when computing RSA with an operand size of 1024 bits, ROS is equal to 33 words, or 1056 bits.
  - EOS (ECC operand size): data size is  $(ecc\_size/32+1)$  words, with  $ecc\_size$  equal to the chosen prime modulus length. For example, when computing ECC with an operand size of 192 bits, EOS is equal to 7 words, or 224 bits.

**Note:** *Fractional results for above formulas are rounded up to the nearest integer since PKA core processes 32-bit words.*

**Note:** *The maximum ROS is 99 words (3136-bit max exponent size), while the maximum EOS is 21 words (640-bit max operand size).*

- The column indicated with Storage in the following tables indicates either the Register PKA\_CR, or the address offset within the PKA, located in the PKA RAM area (starting from 0x400). To recover the physical address of an operand the application must add to the indicated offset the base address of the PKA.
- About writing parameters (“IN” direction)
  - When elements are written as input in the PKA memory, an additional word with all bits equal to zero must be added. As an example, when ECC P256 is used and when loading an input (represented on 256 bits or 8 words), an additional word is expected by the PKA and it has to be filled with zeros.
  - About endianess, for example to prepare the operation ECC Fp scalar multiplication, when application writes  $x_p$  coordinate for an ECC P256 curve (EOS= 9 words) the least significant bit is to be placed in bit 0 at address offset 0x55C; the most significant bit is to be placed in bit 31 of address offset 0x578. Then, as mentioned above, word 0x00000000 should also be written at address offset 0x57C.
  - Unless indicated otherwise all operands in the tables are integers.
- About PKA outputs (“OUT” direction)
  - Unless specified in the tables PKA does not provide an error output when a wrong parameter is written by the application.

**Caution:** Validity of all input parameters to the PKA must be checked before issuing any PKA operation. Indeed, the PKA assumes that all input parameters are valid and consistent with each other.

### 23.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter ( $R^2 \bmod n$ ) used by PKA to convert operands into the Montgomery residue system representation.

**Note:** *This operation can also be used with ECC curves. In this case prime modulus length and EOS size must be used.*

Operation instructions for Montgomery parameter computation are summarized in [Table 129](#).

**Table 129. Montgomery parameter computation**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x01	PKA_CR	6 bits
	Modulus length	(In bits, $0 \leq \text{value} < 3136\text{bits}$ )	RAM@0x404	32 bits
	Modulus value n	(Odd integer only, $n < 2^{3136}$ )	RAM@0xD5C	ROS
OUT	Result: $R^2 n$	-	RAM@0x594	

### 23.4.3 Modular addition

Modular addition operation consists in the computation of  $A + B \bmod n$ . Operation instructions are summarized in [Table 130](#).

**Table 130. Modular addition**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0E	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	( $0 \leq A < n$ )	RAM@0x8B4	ROS
	Operand B	( $0 \leq B < n$ )	RAM@0xA44	
	Modulus value n	( $n < 2^{3136}$ )	RAM@0xD5C	
OUT	Result: $A+B \bmod n$	( $0 \leq \text{result} < n$ )	RAM@0xBD0	

### 23.4.4 Modular subtraction

Modular subtraction operation consists in the following computations:

- If  $A \geq B$  result equals  $A - B \bmod n$
- If  $A < B$  result equals  $A + n - B \bmod n$

Operation instructions are summarized in [Table 131](#).

**Table 131. Modular subtraction**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0F	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	( $0 \leq A < n$ )	RAM@0x8B4	ROS
	Operand B	( $0 \leq B < n$ )	RAM@0xA44	
	Modulus value n	( $n < 2^{3136}$ )	RAM@0xD5C	
OUT	Result: $A-B \bmod n$	( $0 \leq \text{result} < n$ )	RAM@0xBD0	

### 23.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications the PKA accelerates multiplication which has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication  $A \times B \bmod n$

The method to perform above operations are described below. Note that “x” function is this operation, and A, B, C operands are in the natural domain.

1. Inward (or outward) conversion into (or from) Montgomery domain
  - a) Let's assume A is an integer in the natural domain
 

Compute  $r2modn$  using [Montgomery parameter computation](#)

Result  $AR = A \times r2modn \bmod n$  is A in the Montgomery domain
  - b) Let's assume  $BR$  is an integer in the Montgomery domain
 

Result  $B = BR \times 1 \bmod n$  is B in the natural domain

Similarly, above value  $AR$  computed in a) can be converted into the natural domain by computing  $A = AR \times 1 \bmod n$
2. Simple modular multiplication  $A \times B \bmod n$ 
  - a) Compute  $r2modn$  using [Montgomery parameter computation](#)
  - b) Compute  $AR = A \times r2modn \bmod n$ . Output is in the Montgomery domain
  - c) Compute  $AB = AR \times B \bmod n$ . Output is in natural domain
3. Multiple modular multiplication  $A \times B \times C \bmod n$ 
  - a) Compute  $r2modn$  using [Montgomery parameter computation](#)
  - b) Compute  $AR = A \times r2modn \bmod n$ . Output is in the Montgomery domain
  - c) Compute  $BR = B \times r2modn \bmod n$ . Output is in the Montgomery domain
  - d) Compute  $ABR = AR \times BR \bmod n$ . Output is in the Montgomery domain
  - e) Compute  $CR = C \times r2modn \bmod n$ . Output is in the Montgomery domain
  - f) Compute  $ABCR = ABR \times CR \bmod n$ . Output is in the Montgomery domain
  - g) (optional) Repeat the two steps above if more operands need to be multiplied
  - h) Compute  $ABC = ABCR \times 1 \bmod n$  to retrieve the result in natural domain

Operation instructions for Montgomery multiplication are summarized in [Table 132](#).

**Table 132. Montgomery multiplication**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x10	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	$(0 \leq A < n)$	RAM@0x8B4	ROS
	Operand B	$(0 \leq B < n)$	RAM@0xA44	
	Modulus value n	(Odd integer only, $n < 2^{3136}$ )	RAM@0xD5C	
OUT	Result: $A \times B \bmod n^{(1)}$	-	RAM@0xBD0	

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples [2](#) and [3](#)).

### 23.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of  $A^e \bmod n$ .

Operation instructions for modular exponentiation are summarized in [Table 133](#) (normal mode) and in [Table 134](#) (fast mode). Fast mode usage is explained in [Section 23.3.6](#).

**Table 133. Modular exponentiation (normal mode)**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x00	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	32 bits
	Operand length	(in bits, not null)	RAM@0x404	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xA44	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xBD0	
	Modulus value n	(Odd integer only, n < $2^{3136}$ )	RAM@0xD5C	
OUT	Result: $A^e \bmod n$	(0 ≤ result < n)	RAM@0x724	

**Table 134. Modular exponentiation (fast mode)**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x02	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	32 bits
	Operand length	(in bits, not null)	RAM@0x404	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xA44	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xBD0	
	Modulus value n	(Odd integer only, n < $2^{3136}$ )	RAM@0xD5C	
IN/OUT	Montgomery param R2 mod n	(mandatory)	RAM@0x594	
OUT	Result: $A^e \bmod n$	(0 ≤ result < n)	RAM@0x724	

### 23.4.7 Modular inversion

Modular inversion operation consists in the computation of multiplicative inverse  $A^{-1} \bmod n$ . If the modulus  $n$  is prime, for all values of  $A$  ( $1 \leq A < n$ ) modular inversion output is valid. If the modulus  $n$  is not prime,  $A$  has an inverse only if the largest common divisor between  $A$  and  $n$  is 1.

If the operand  $A$  is a divisor of the modulus  $n$ , the result is a multiple of a factor of  $n$ .

Operation instructions for modular inversion are summarized in [Table 135](#).

**Table 135. Modular inversion**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x08	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x404	32 bits
	Operand A	(0 ≤ A < n)	RAM@0x8B4	ROS
	Modulus value n	(Odd integer only, n < $2^{3136}$ )	RAM@0xA44	
OUT	Result: $A^{-1} \bmod n$	0 < result < n	RAM@0xBD0	

### 23.4.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of A divided by n. Operation instructions are summarized in [Table 136](#).

**Table 136. Modular reduction**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0D	PKA_CR	6 bits
	Operand length	(In bits, not null)	RAM@0x400	32 bits
	Modulus length	(In bits, 8 < value < 3136)	RAM@0x404	
	Operand A	( $0 \leq A < 2^{3136}$ )	RAM@0x8B4	ROS
	Modulus value n	(Odd integer only, $n < 2^{3136}$ )	RAM@0xA44	
OUT	Result A mod n	( $0 < \text{result} < n$ )	RAM@0xBD0	

### 23.4.9 Arithmetic addition

Arithmetic addition operation consists in the computation of A + B. Operation instructions are summarized in [Table 137](#).

**Table 137. Arithmetic addition**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x09	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	( $0 \leq A < 2^M$ )	RAM@0x8B4	
	Operand B	( $0 \leq B < 2^M$ )	RAM@0xA44	ROS
OUT	Result: A+B	( $0 \leq \text{result} < 2^{M+1}$ )	RAM@0xBD0	ROS + 1

### 23.4.10 Arithmetic subtraction

Arithmetic subtraction operation consists in the following computations:

- If  $A \geq B$  result equals  $A - B$
- If  $A < B$  and  $M/32$  residue is  $> 0$  result equals  $A + 2^{\text{int}(M/32)*32+1} - B$
- If  $A < B$  and  $M/32$  residue is 0 result equals  $A + 2^{\text{int}(M/32)*32} - B$

Operation instructions are summarized in [Table 138](#).

**Table 138. Arithmetic subtraction**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0A	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	( $0 \leq A < 2^M$ )	RAM@0x8B4	
	Operand B	( $0 \leq B < 2^M$ )	RAM@0xA44	ROS
OUT	Result: A-B	( $0 \leq \text{result} < 2^M$ )	RAM@0xBD0	

### 23.4.11 Arithmetic multiplication

Arithmetic multiplication operation consists in the computation of  $A \times B$ . Operation instructions are summarized in [Table 139](#).

**Table 139. Arithmetic multiplication**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0B	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	$(0 \leq A < 2^M)$	RAM@0x8B4	ROS
	Operand B	$(0 \leq B < 2^M)$	RAM@0xA44	
OUT	Result: $A \times B$	$(0 \leq \text{result} < 2^M)$	RAM@0xBD0	2xROS

### 23.4.12 Arithmetic comparison

Arithmetic comparison operation consists in the following computation:

- If  $A=B$  then result=0x0
- If  $A>B$  then result=0x1
- If  $A<B$  then result=0x2

Operation instructions for arithmetic comparison are summarized in [Table 140](#).

**Table 140. Arithmetic comparison**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0C	PKA_CR	6 bits
	Operand length M	(In bits, not null)	RAM@0x404	32 bits
	Operand A	$(0 \leq A < 2^M)$	RAM@0x8B4	ROS
	Operand B	$(0 \leq B < 2^M)$	RAM@0xA44	
OUT	Result $A=B$ or $A>B$ or $A<B$	0x0, 0x1 or 0x02	RAM@0xBD0	32 bits

### 23.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries like OpenSSL RSA use the following optimization for decryption and signing based on the Chinese remainder theorem (CRT):

- $p$  and  $q$  are precomputed primes, stored as part of the private key
- $d_p = d \bmod (p - 1)$
- $d_Q = d \bmod (q - 1)$  and
- $q_{\text{inv}} = q^{-1} \bmod p$

These values allow the recipient to compute the exponentiation  $m = A^d \pmod{pq}$  more efficiently as follows:

- $m_1 = A^{dP} \pmod{p}$
- $m_2 = A^{dQ} \pmod{p}$
- $h = q_{\text{inv}} (m_1 - m_2) \pmod{p}$ , with  $m_1 > m_2$
- $m = m_2 + hq$

Operation instructions for computing CRT exponentiation  $A^d \pmod{pq}$  are summarized in [Table 141](#).

**Table 141. CRT exponentiation**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x404	32 bits
IN	Operand $d_P$	$(0 \leq d_P < 2^{M/2})$	RAM@0x65C	ROS/2
	Operand $d_Q$	$(0 \leq d_Q < 2^{M/2})$	RAM@0xBD0	
	Operand $q_{\text{inv}}$	$(0 \leq q_{\text{inv}} < 2^{M/2})$	RAM@0x7EC	
	Prime $p^{(1)}$	$(0 \leq p < 2^{M/2})$	RAM@0x97C	
	Prime $q^{(1)}$	$(0 \leq q < 2^{M/2})$	RAM@0xD5C	
IN	Operand A	$(0 \leq A < 2^{M/2})$	RAM@0xEEC	ROS
OUT	Result: $A^d \pmod{pq}$	$(0 \leq \text{result} < pq)$	RAM@0x724	

1. Must be different from 2.

#### 23.4.14 Point on elliptic curve Fp check

This operation consists in checking whether a given point  $P(x, y)$  satisfies or not the curves over prime fields equation  $y^2 = (x^3 + ax + b) \pmod{p}$ , where  $a$  and  $b$  are elements of the curve.

Operation instructions for point on elliptic curve Fp check are summarized in [Table 142](#).

Table 142. Point on elliptic curve Fp check

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x28	PKA_CR	6 bits
	Modulus length	(In bits, not null, 8 < value < 640)	RAM@0x404	32 bits
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x408	
	Curve coefficient  a	(Absolute value,  a  < p)	RAM@0x40C	EOS
	Curve coefficient b	( b  < p)	RAM@0x7FC	
	Curve modulus value p	(Odd integer prime, 0 < p < 2640)	RAM@0x460	
	Point P coordinate x	(x < p)	RAM@0x55C	
	Point P coordinate y	(y < p)	RAM@0x5B0	
OUT	Result: P on curve	0x0: point on curve Not 0x0: point not on curve	RAM@0x400	32 bits

### 23.4.15 ECC Fp scalar multiplication

This operation consists in the computation of a  $k \times P$  ( $x_P, y_P$ ), where  $P$  is a point on a curve over prime fields and “x” is the elliptic curve scalar point multiplication. Result of the computation is a point that belongs to the same curve or a point at infinity.

Operation instructions for ECC Fp scalar multiplication are summarized in [Table 143](#) (normal mode) and [Table 144](#) (fast mode). Fast mode usage is explained in [Section 23.3.6](#).

Table 143. ECC Fp scalar multiplication

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x20	PKA_CR	6 bits
IN	Scalar multiplier k length	(In bits, not null, 8 < value < 640)	RAM@0x400	32 bits
	Modulus length	(In bits, not null, 8 < value < 640)	RAM@0x404	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x408	
IN	Curve coefficient  a	(Absolute value,  a  < p)	RAM@0x40C	EOS
	Curve modulus value p	(Odd integer prime, 0 < p < $2^{640}$ )	RAM@0x460	
	Scalar multiplier k	( $0 \leq k < 2^{640}$ )	RAM@0x508	
	Point P coordinate $x_P$	( $x < p$ )	RAM@0x55C	
	Point P coordinate $y_P$	( $y < p$ )	RAM@0x5B0	
OUT	Result: $k \times P$ coordinate x	(result < p)	RAM@0x55C	649/1530
	Result: $k \times P$ coordinate y	(result < p)	RAM@0x5B0	

Table 144. ECC Fp scalar multiplication (Fast Mode)

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x22	PKA_CR	6 bits
IN	Scalar multiplier k length	(In bits, not null, 8 < value < 640)	RAM@0x400	32 bits
	Modulus length	(In bits, not null, 8 < value < 640)	RAM@0x404	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x408	
IN	Curve coefficient $ a $	(Absolute value, $ a  < p$ )	RAM@0x40C	EOS
	Curve modulus value $p$	(Odd integer prime, $0 < p < 2^{640}$ )	RAM@0x460	
	Scalar multiplier k	( $0 \leq k < 2^{640}$ )	RAM@0x508	
	Point P coordinate $x_P$	( $x < p$ )	RAM@0x55C	
	Point P coordinate $y_P$	( $y < p$ )	RAM@0x5B0	
IN	Montgomery parameter $R^2 \bmod p$	(mandatory)	RAM@0x4B4	
OUT	Result: k x P coordinate x	(result < p)	RAM@0x55C	
	Result: k x P coordinate y	(result < p)	RAM@0x5B0	

When performing this operation following special cases should be noted:

- For  $k = 0$ , this function returns a point at infinity, that is  $(0, 0)$  if curve parameter b is nonzero and  $(0, 1)$  otherwise. For  $k$  different from 0 it might happen that a point at infinity is returned. When the application detects this behavior a new computation should be carried out.
- For  $k < 0$  (i.e. a negative scalar multiplication is required) multiplier absolute value  $k = |-k|$  should be provided to the PKA. After the computation completion, the formula  $-P = (x, -y)$  can be used to compute the y coordinate of the effective final result (the x coordinate remains the same).

### 23.4.16 ECDSA sign

ECDSA signing operation (outlined in [Section 23.3.5](#)) is summarized in [Table 145](#) (input parameters) and in [Table 146](#) (output parameters).

The application should check if the output error is equal to zero, if it is different from zero a new  $k$  should be generated and the ECDSA sign operation should be repeated.

Table 145. ECDSA sign - Inputs

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x24	PKA_CR	6 bits
	Curve prime order n length	(in bits, not null)	RAM@0x400	32 bits
	Curve modulus p length	(in bits, 8 < value < 640)	RAM@0x404	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x408	
	Curve coefficient  a	(Absolute value, $ a  < p$ )	RAM@0x40C	EOS
	Curve modulus value $p$	(Odd integer prime, $0 < p < 2^{640}$ )	RAM@0x460	
	Integer $k^{(1)}$	$(0 \leq k < 2^{640})$	RAM@0x508	
	Curve base point G coordinate x	$(x < p)$	RAM@0x55C	
	Curve base point G coordinate y	$(y < p)$	RAM@0x5B0	
	Hash of message z	$(z < 2M)$	RAM@0xDE8	
Private key d		(positive integer)	RAM@0xE3C	
Curve prime order n		(integer prime)	RAM@0xE94	

1. This integer is usually a cryptographically secure random number, but in some cases k could be deterministically generated.

Table 146. ECDSA sign - Outputs

Parameters with direction		Value (Note)	Storage	Size
OUT	Signature part r	$(0 < r < n)$	RAM@0x700	EOS
	Signature part s	$(0 < s < n)$	RAM@0x754	
ERROR	Result of signature		– 0x0: no error – 0x1: signature part r is equal to 0 – 0x2: signature part s is equal to 0	RAM@0xEE8 32 bits

Note: *If error output is different from zero the content of the PKA memory should be cleared to avoid leaking information about the private key.*

### Extended ECDSA support

PKA also supports Extended ECDSA signature, for which the inputs and the outputs have the same ECDSA signature (Table 145 and Table 146, respectively), with the addition of the coordinates of the point  $kG$ . This extra output is defined in Table 147.

**Table 147. Extended ECDSA sign (extra outputs)**

Parameters with direction		Value (Note)	Storage	Size
OUT	Curve point kG coordinate $x_1$	$(0 \leq x_1 < p)$	RAM@0x103C	EOS
	Curve point kG coordinate $y_1$	$(0 \leq y_1 < p)$	RAM@0x1090	

### 23.4.17 ECDSA verification

ECDSA verification operation (outlined in [Section 23.3.5](#)) is summarized in [Table 148](#) (input parameters) and [Table 149](#) (output parameters).

The application should check if the output error is equal to zero, if it is different from zero, the signature is not verified.

**Table 148. ECDSA verification (inputs)**

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x26	PKA_CR	6 bits
	Curve prime order $n$ length	(In bits, not null)	RAM@0x404	32 bits
	Curve modulus $p$ length	(In bits, not null, $8 < \text{value} < 640$ )	RAM@0x4B4	
	Curve coefficient $a$ sign	0x0: positive 0x1: negative	RAM@0x45C	
	Curve coefficient $ a $	(Absolute value, $ a  < p$ )	RAM@0x460	EOS
	Curve modulus value $p$	(Odd integer prime, $0 < p < 2^{640}$ )	RAM@0x4B8	
	Curve base point $G$ coordinate $x$	$(x < p)$	RAM@0x5E8	
	Curve base point $G$ coordinate $y$	$(y < p)$	RAM@0x63C	
	Public-key curve point $Q$ coordinate $x_Q$	$(x_Q < p)$	RAM@0xF40	
	Public-key curve point $Q$ coordinate $y_Q$	$(y_Q < p)$	RAM@0xF94	
	Signature part $r$	$(0 < r < n)$	RAM@0x1098	
	Signature part $s$	$(0 < s < n)$	RAM@0xA44	
	Hash of message $z$	$(z < 2^M)$	RAM@0xFE8	
	Curve prime order $n$	(integer prime)	RAM@0xD5C	

**Table 149. ECDSA verification (outputs)**

Parameters with direction		Value (Note)	Storage	Size
OUT	Result: ECDSA verify	0x0: valid signature Not 0x0: invalid signature	RAM@0x5B0	32 bits

## 23.5 Example of configurations and processing times

### 23.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation  $y^2 = x^3 + ax + b \pmod{p}$ .

**Note:** *Binary curves, Edwards curves and Curve25519 are not supported by the PKA. The maximum supported operand size for ECC operations is 640 bits.*

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer  $p$ , used as the modulus for all point arithmetic in the finite field  $GF(p)$
- the (usually prime) integer  $n$ , the order of the group generated by  $G$ , defined below
- the base point of the curve  $G$ , defined by its coordinates  $(G_x, G_y)$
- the integers  $a$  and  $b$ , coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define  $a$  as negative, PKA supports two representations:

1. **a defined as  $p-|a|$**  in the finite field  $GF(p)$ , for example **p-3**:  
 Curve coefficient  $p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF$   
 Curve coefficient  $a$  sign= 0x0 (positive)  
 Curve coefficient  $a = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFC$
2. **a defined as negative**, for example **-3**:  
 Curve coefficient  $p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF$   
 Curve coefficient  $a$  sign= 0x1 (negative)  
 Curve coefficient  $a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000003$

*Table 150* summarizes the family of curves supported by PKA for ECC operations.

**Table 150. Family of supported curves for ECC operations**

Curve name	Standard	Reference
P-192	NIST	<i>Digital Signature Standard (DSS)</i> , NIST FIPS 186-4
P-224		
P-256		
P-384		
P-521		

Table 150. Family of supported curves for ECC operations (continued)

Curve name	Standard	Reference	
brainpoolP224r1, brainpoolP224t1	IETF	<ul style="list-style-type: none"> <li>– <i>Brainpool Elliptic Curves</i>, IETF RFC 5639</li> <li>– <i>Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry</i>, IETF RFC 6932</li> </ul>	<a href="https://tools.ietf.org">https://tools.ietf.org</a>
brainpoolP256r1, brainpoolP256t1			
brainpoolP320r1, brainpoolP320t1			
brainpoolP384r1, brainpoolP384t1			
brainpoolP512r1, brainpoolP512t1			
secp192k1, secp192r1	SEC	<i>Standards for Efficient Cryptography SEC 2 curves</i>	<a href="https://www.secg.org">https://www.secg.org</a>
secp224k1, secp224r1			
secp256k1, secp256r1			
secp384r1			
secp521r1			
Recommended curve parameters for public key cryptographic algorithm SM2	OSCCA	<ul style="list-style-type: none"> <li>– <i>Public key cryptographic algorithm SM2 based on elliptic curves</i>, Organization of State Commercial Administration of China OSCCA SM2, December 2010</li> <li>– <i>Digital signatures - Part 3 Discrete logarithm based mechanisms</i>, ISO/IEC 14888-3, November 2018</li> </ul>	

### 23.5.2 Computation times

The following tables summarize the PKA computation times, expressed in clock cycles.

**Table 151. Modular exponentiation computation times**

Exponent length (in bits)	Mode	Modulus length (in bits)		
		1024	2048	3072
3	Normal	304000	814000	1728000
	Fast	46000	164000	356000
17	Normal	326000	896000	1910000
	Fast	68000	246000	534000
$2^{16} + 1$	Normal	416000	1222000	2616000
	Fast	158000	572000	1244000
1024	Normal	11664000	-	-
	Fast	11280000	-	-
	CRT <sup>(1)</sup>	3546000	-	-
2048	Normal	-	83834000	-
	Fast	-	82046000	-
	CRT <sup>(1)</sup>	-	23468000	-
3072	Normal	-	-	274954000
	Fast	-	-	273522000
	CRT <sup>(1)</sup>	-	-	73378000

1. CRT stands for chinese remainder theorem optimization (MODE bitfield = 0x07).

**Table 152. ECC scalar multiplication computation times<sup>(1)</sup>**

Mode	Modulus length (in bits)						
	160	192	256	320	384	512	521
Normal	1634000	2500000	4924000	8508000	13642000	28890000	33160000
Fast	1630000	2494000	4916000	8494000	13614000	28842000	33158000

1. These times depend on the number of "1"s included in the scalar parameter.

**Table 153. ECDSA signature average computation times<sup>(1) (2)</sup>**

Modulus length (in bits)						
160	192	256	320	384	512	521
1760000	2664000	5249000	9016000	14596000	30618000	35540000

- These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
- The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

**Table 154. ECDSA verification average computation times**

Modulus length (in bits)						
160	192	256	320	384	512	521
3500000	5350000	10498000	18126000	29118000	61346000	71588000

**Table 155. Point on elliptic curve Fp check average computation times**

Modulus length (in bits)					
160	192	256	320	384	512
10800	14200	20400	31000	49600	82400

**Table 156. Montgomery parameters average computation times<sup>(1)</sup>**

Modulus length (in bits)									
160	192	256	320	384	512	521	1024	2048	3072
4518	7846	11848	14902	21682	35012	64000	119536	466146	1104642

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

## 23.6 PKA interrupts

There are three individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. access to unmapped address (ADDRERRF), see [Section 23.3.7](#)
2. PKA RAM access while PKA operation is in progress (RAMERRF), see [Section 23.3.7](#)
3. PKA end of operation (PROCENDF)

The three interrupt sources are connected to the same global interrupt request signal pka\_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the [PKA control register \(PKA\\_CR\)](#). Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA\_SR), and it is cleared in PKA\_CLRFR register.

[Table 157](#) gives a summary of the available features.

**Table 157. PKA interrupt requests**

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
PKA	Access to unmapped address error	ADDRERRF	ADDRERRIE	Set ADDRERRFC bit
	PKA RAM access error	RAMERRF	RAMERRIE	Set RAMERRFC bit
	PKA end of operation	PROCENDF	PROCENDIE	Set PROCENDFC bit

## 23.7 PKA registers

### 23.7.1 PKA control register (PKA\_CR)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

Bit 20 **ADDRERRIE**: Address error interrupt enable

- 0: No interrupt is generated when ADDRERRF flag is set in PKA\_SR.
- 1: An interrupt is generated when ADDRERRF flag is set in PKA\_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable

- 0: No interrupt is generated when RAMERRF flag is set in PKA\_SR.
- 1: An interrupt is generated when RAMERRF flag is set in PKA\_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable

- 0: No interrupt is generated when PROCENDF flag is set in PKA\_SR.
- 1: An interrupt is generated when PROCENDF flag is set in PKA\_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code

- 000000: Montgomery parameter computation then modular exponentiation
- 000001: Montgomery parameter computation only
- 000010: Modular exponentiation only (Montgomery parameter must be loaded first)
- 100000: Montgomery parameter computation then ECC scalar multiplication
- 100010: ECC scalar multiplication only (Montgomery parameter must be loaded first)
- 100100: ECDSA sign
- 100110: ECDSA verification
- 101000: Point on elliptic curve Fp check
- 000111: RSA CRT exponentiation
- 001000: Modular inversion
- 001001: Arithmetic addition
- 001010: Arithmetic subtraction
- 001011: Arithmetic multiplication
- 001100: Arithmetic comparison
- 001101: Modular reduction
- 001110: Modular addition
- 001111: Modular subtraction
- 010000: Montgomery multiplication
- Others: Reserved

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **START**: start the operation

Writing 1 to this bit starts the operation which is selected by MODE[5:0], using the operands and data already written to the PKA RAM. This bit is always read as 0.

*Note: START is ignored if PKA is busy.*

Bit 0 **EN**: PKA enable.

0: Disable PKA

1: Enable PKA

*Note: When EN=0 PKA RAM can still be accessed by the application.*

### 23.7.2 PKA status register (PKA\_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ADDR ERRF	RAM ERRF	Res.	PROC ENDF	BUSY										
											r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.											

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

Bit 20 **ADDRERRF**: Address error flag

0: No Address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA\_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA\_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA\_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA End of Operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: PKA operation is in progress

This bit is set to 1 whenever START bit in the PKA\_CR is set. It is automatically cleared when the computation is complete, meaning that PKA RAM can be safely accessed and a new operation can be started.

0: No operation is in progress (default)

1: An operation is in progress

If PKA is started with a wrong opcode the peripheral is busy for a couple of cycles, then it aborts automatically the operation and go back to ready (BUSY bit is set to 0).

Bits 15:0 Reserved, must be kept at reset value.

### 23.7.3 PKA clear flag register (PKA\_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ADDR ERRFC	RAM ERRFC	Res.	PROC ENDFC	Res.										
											w	w		w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.											

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

Bit 20 **ADDRERRFC**: Clear Address error flag

0: No action

1: Clear the ADDRERRF flag in PKA\_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag

0: No action

1: Clear the RAMERRF flag in PKA\_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA End of Operation flag

0: No action

1: Clear the PROCENDF flag in PKA\_SR

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

*Note:* *Reading PKA\_CLRFR returns all 0s.*

### 23.7.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 3576 bytes (max word offset: 0x11F4).

### 23.7.5 PKA register map

Table 158. PKA register map and reset values

Offset	Register name	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PKA_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		Reset value	Res.																															
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		Reset value	Res.																															
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	0	0	0	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	0		

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 24 Advanced-control timer (TIM1)

In this section, “TIMx” should be understood as “TIM1” since there is only one instance of this type of timer for the products to which this reference manual applies.

### 24.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

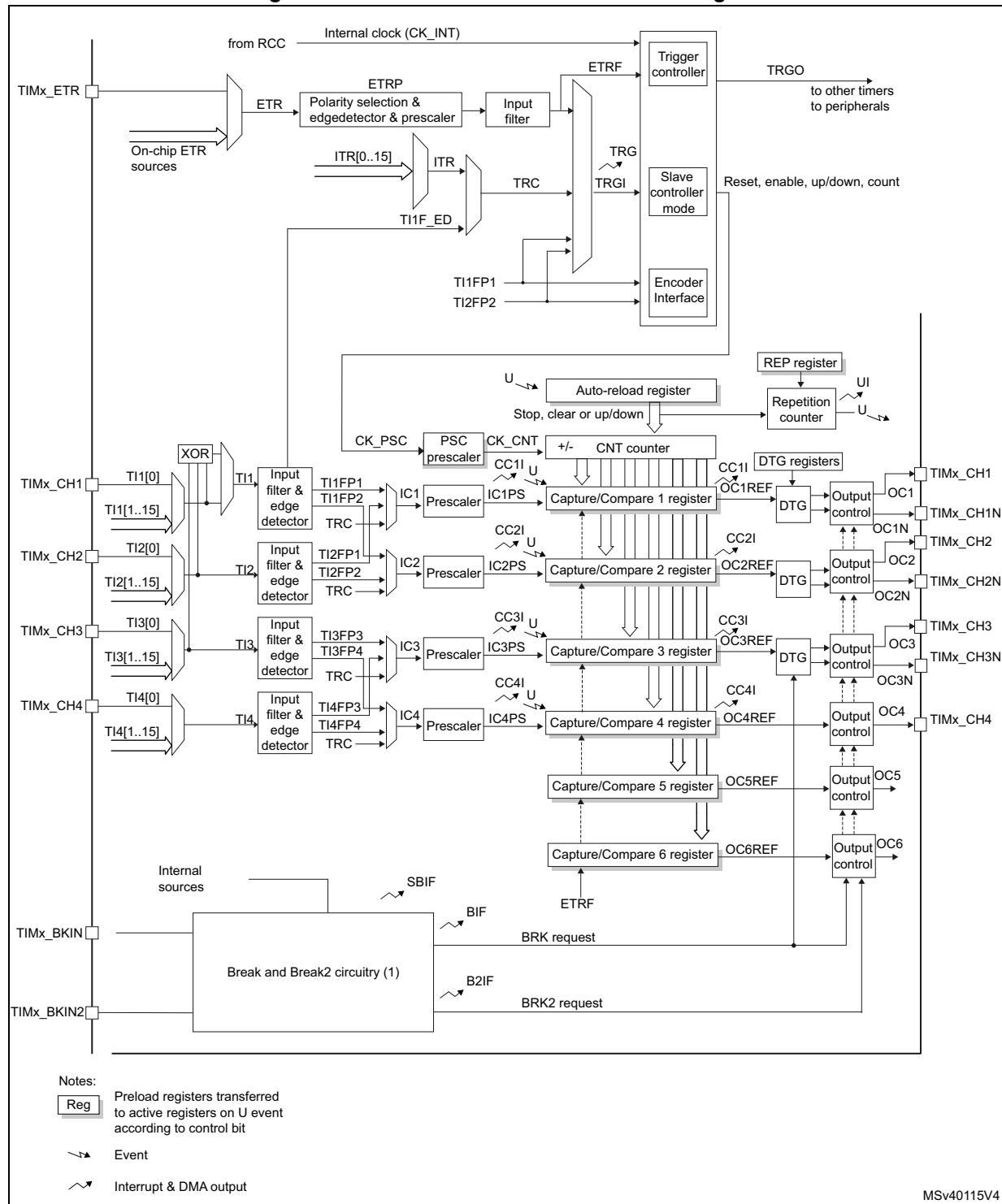
The advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 24.3.26: Timer synchronization](#).

## 24.2 TIM1 main features

TIM1 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  - Input Capture (but channels 5 and 6)
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 144. Advanced-control timer block diagram



1. The internal break event source can be:
  - A clock failure event generated by CSS. For further information on the CSS, refer to [Section 8.2.11: Clock security system \(CSS\) on HSE](#)
  - A PVD output
  - SRAM parity error signal
  - CPU1 Cortex®-M4 LOCKUP (Hardfault) output.
  - COMP<sub>x</sub> output, x = 1,2.

## 24.3 TIM1 functional description

### 24.3.1 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 145* and *Figure 146* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 145. Counter timing diagram with prescaler division change from 1 to 2

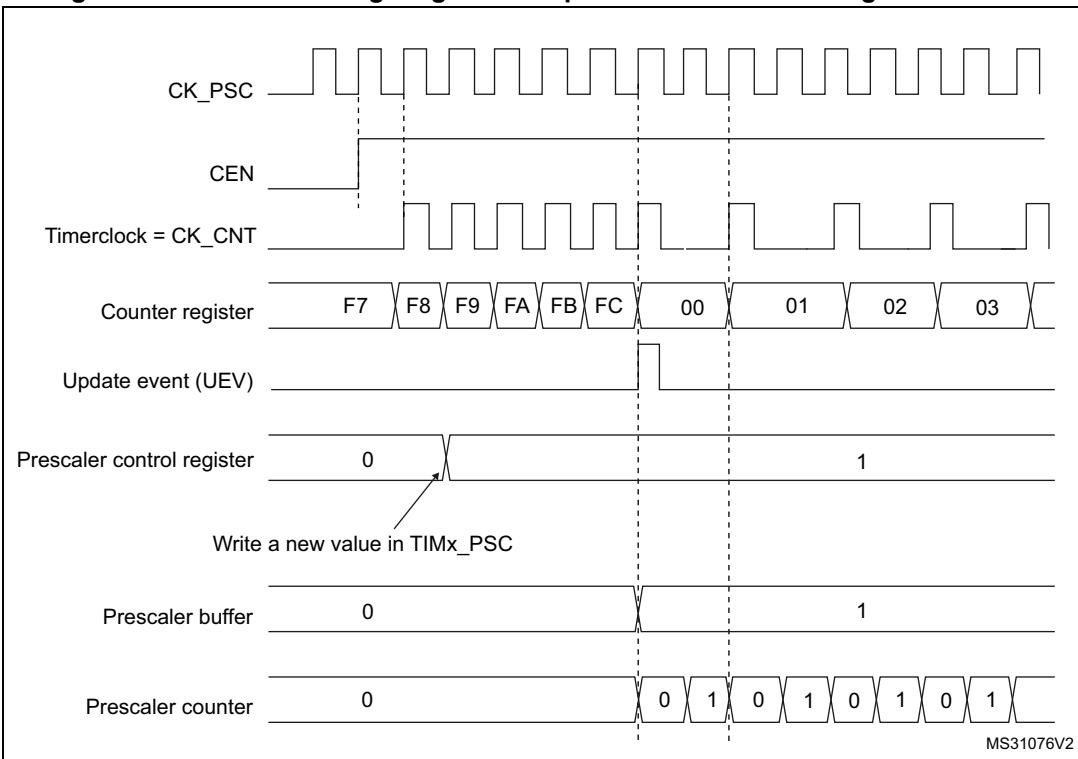
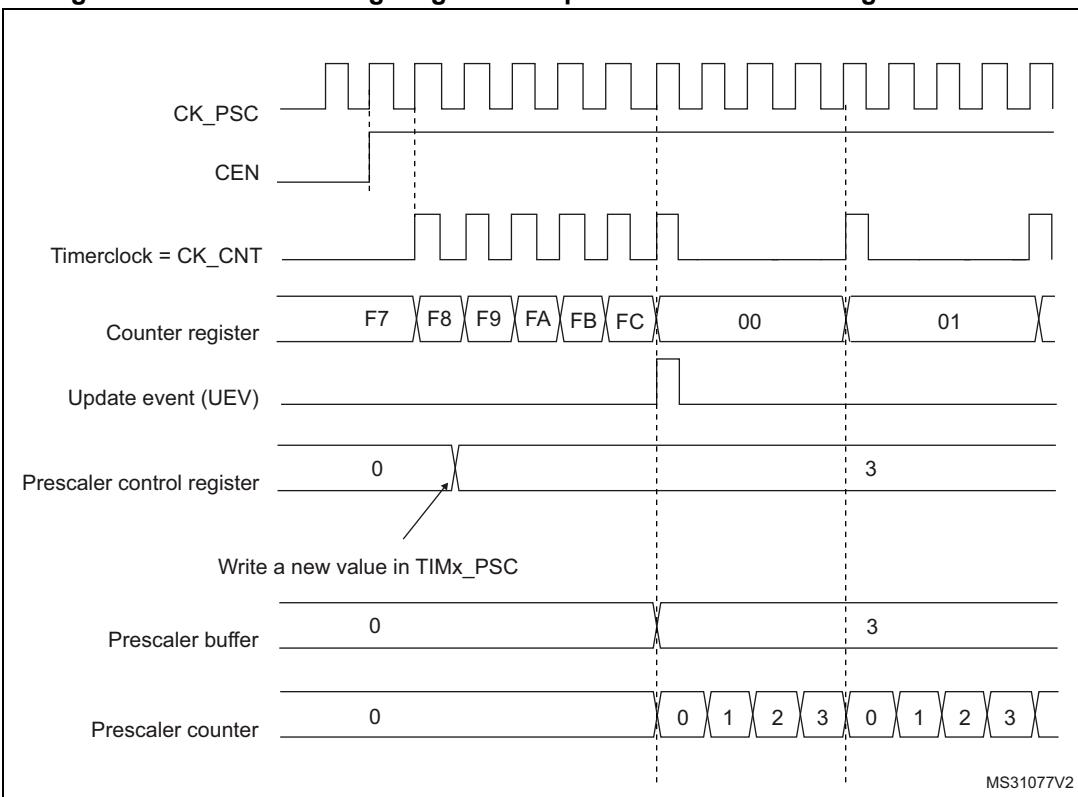


Figure 146. Counter timing diagram with prescaler division change from 1 to 4



### 24.3.2 Counter modes

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx\_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx\_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

Figure 147. Counter timing diagram, internal clock divided by 1

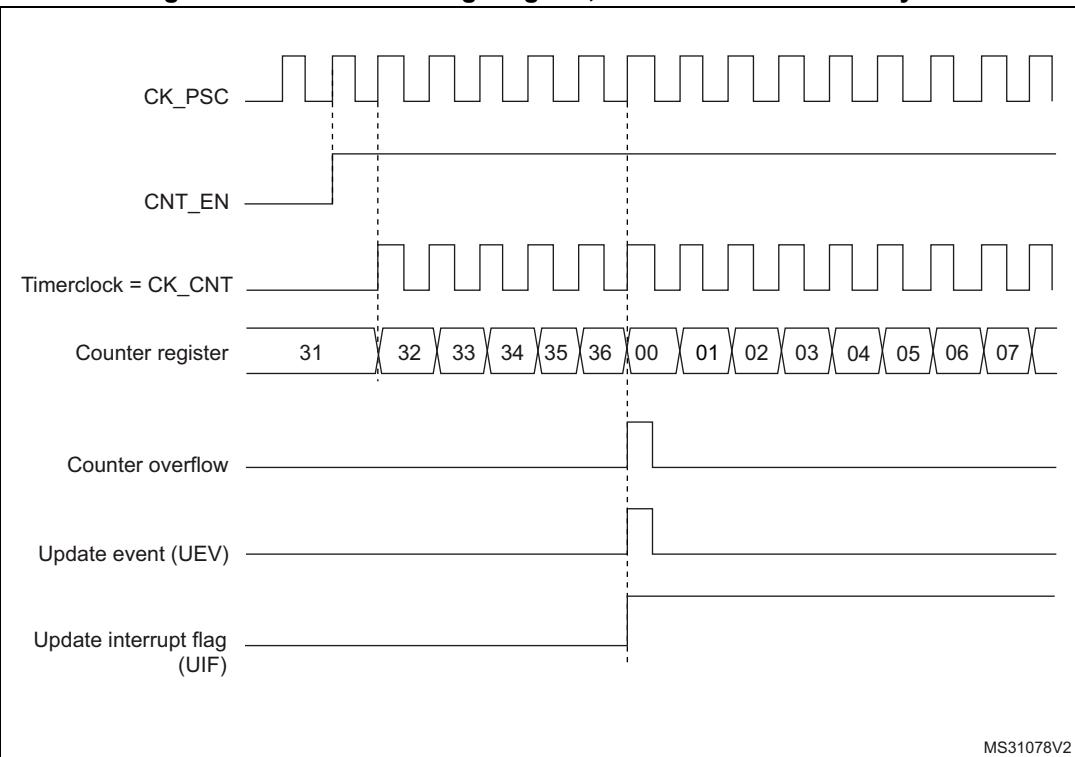


Figure 148. Counter timing diagram, internal clock divided by 2

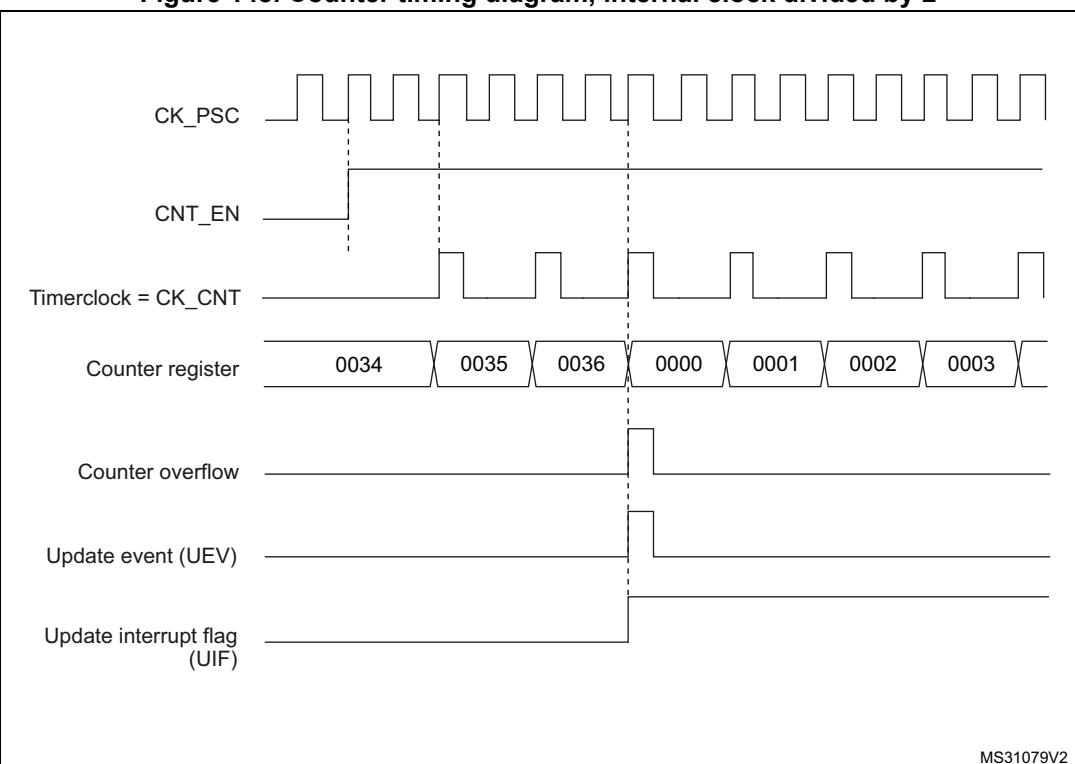


Figure 149. Counter timing diagram, internal clock divided by 4

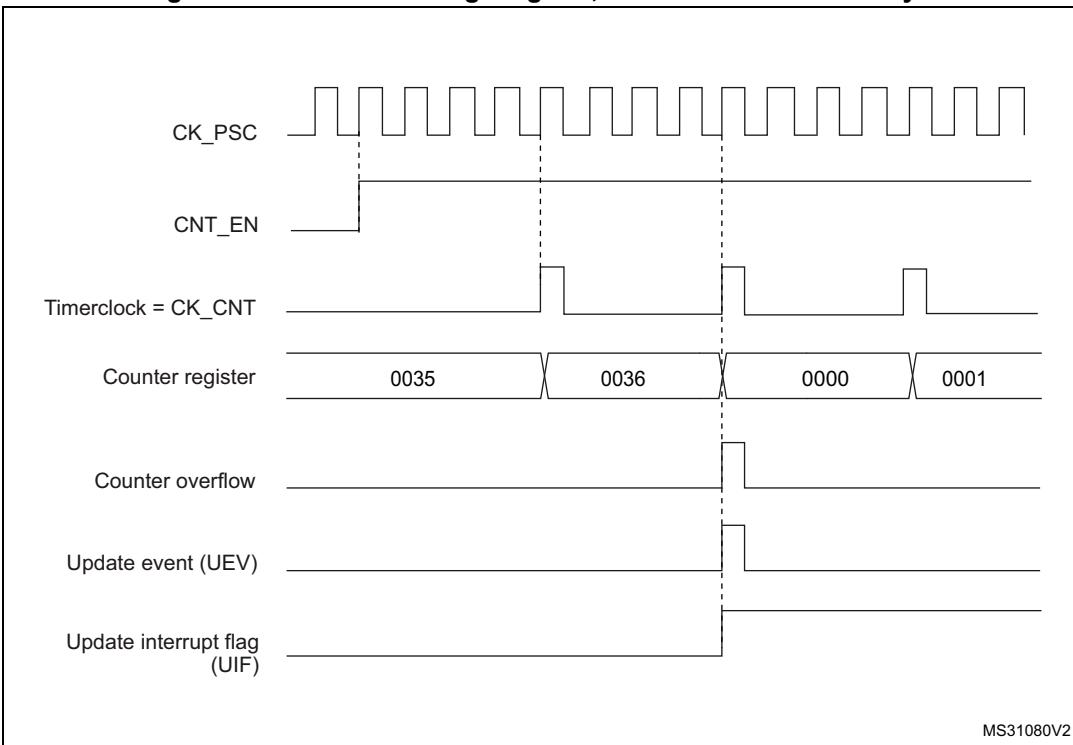
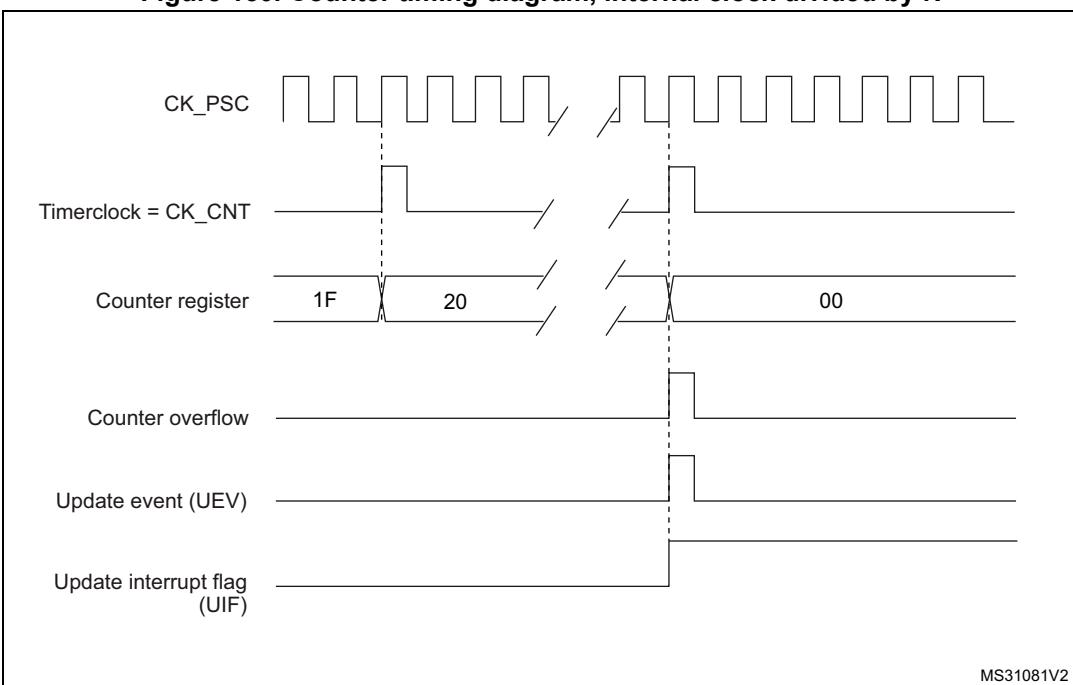
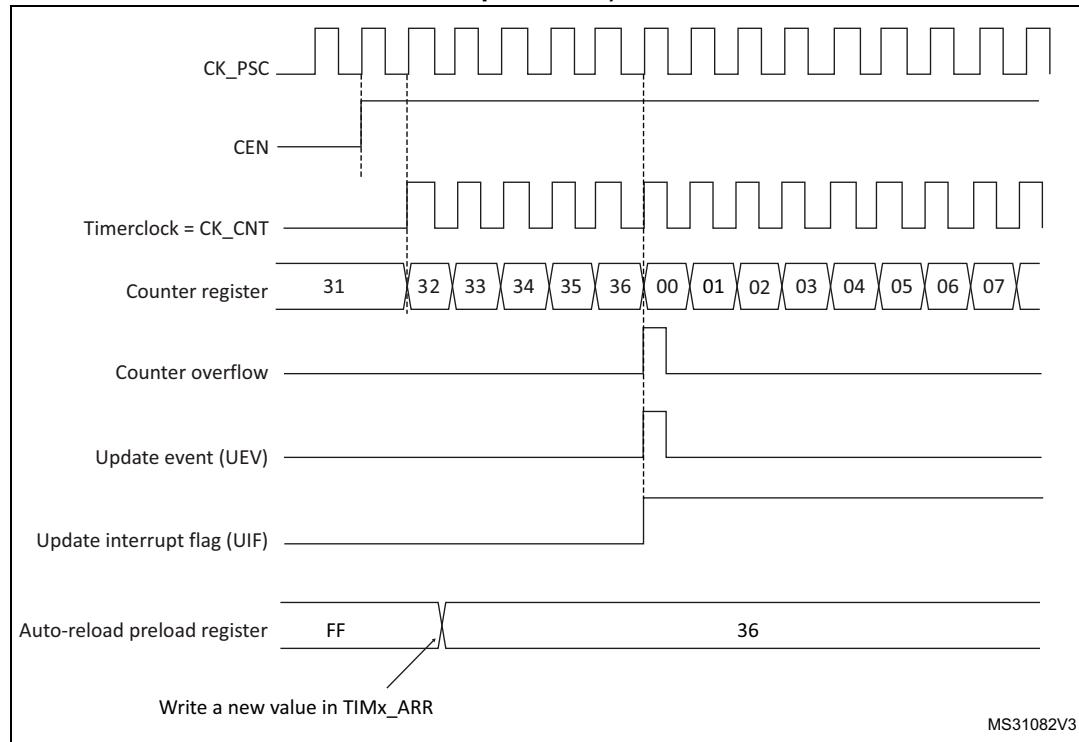


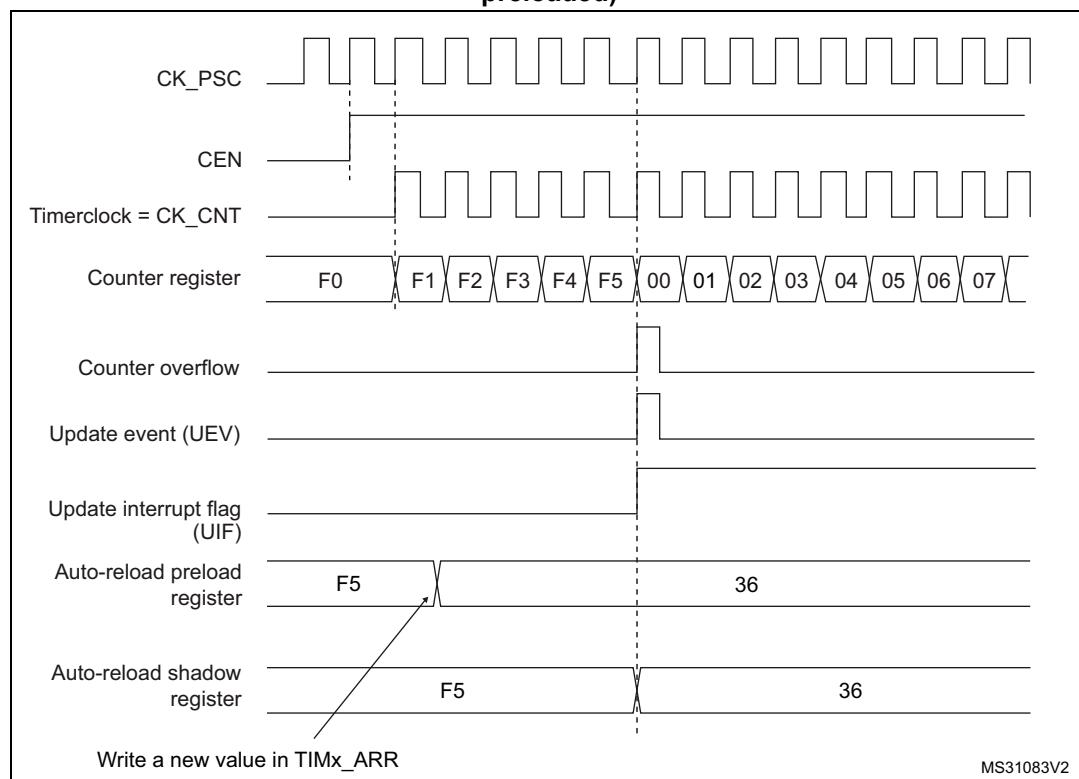
Figure 150. Counter timing diagram, internal clock divided by N



**Figure 151. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)**



**Figure 152. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



## Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx\_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx\_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx\_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

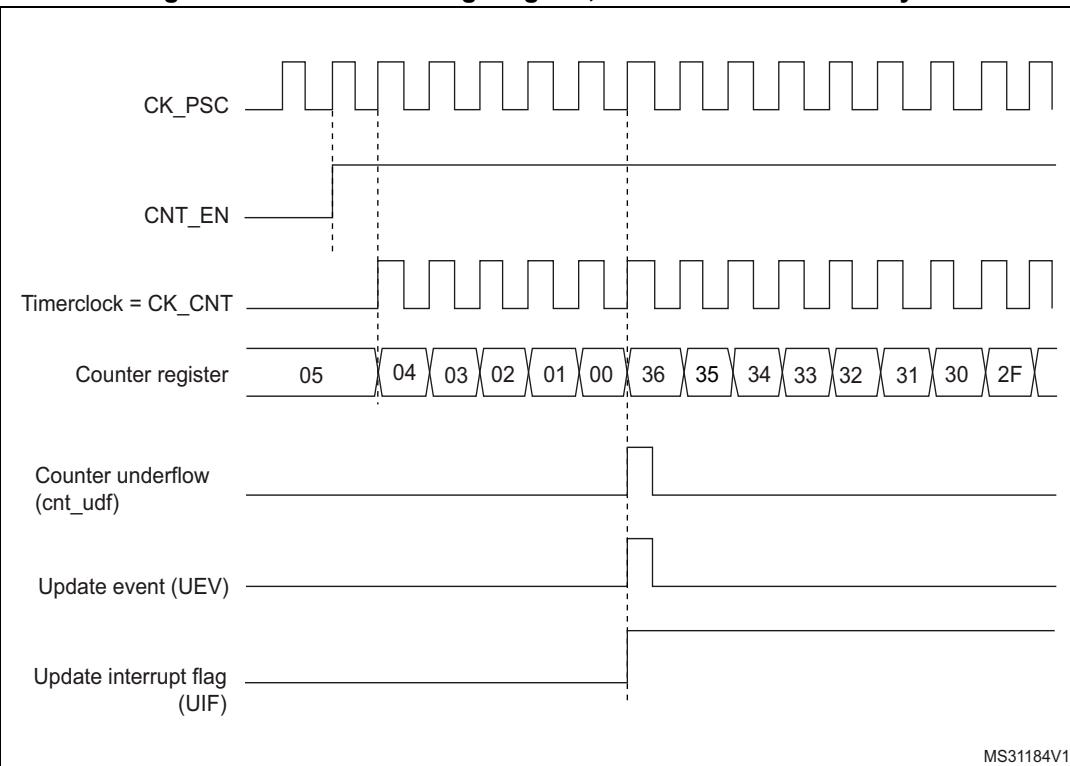
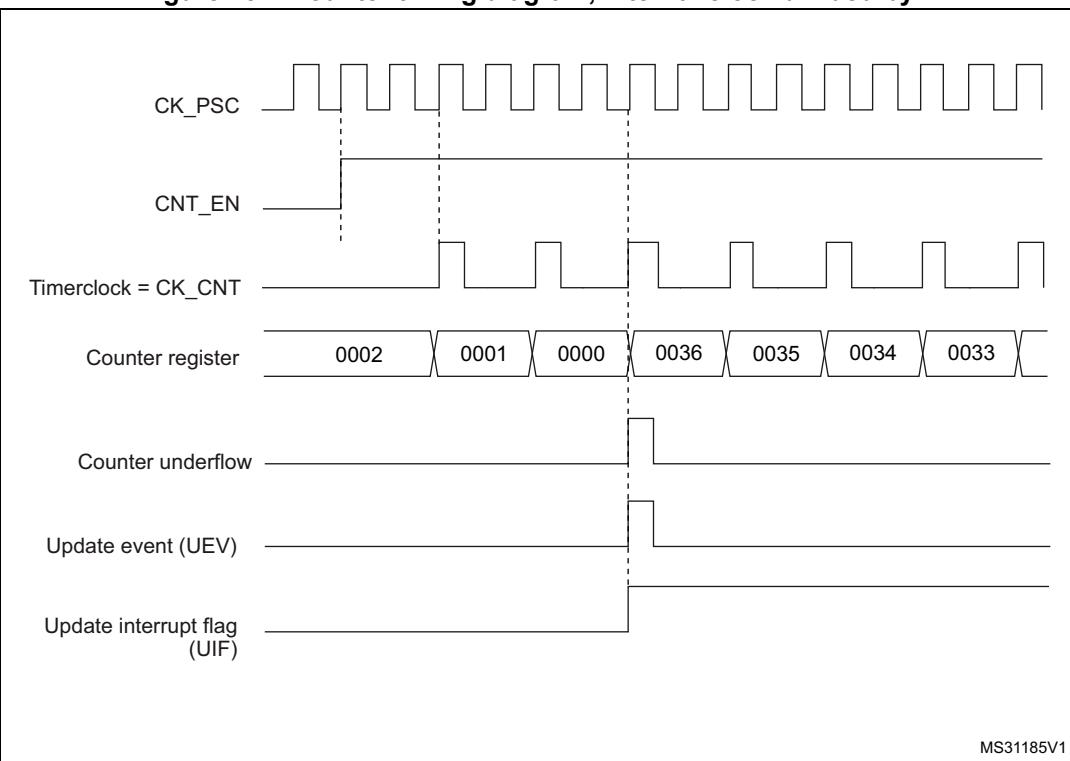
**Figure 153. Counter timing diagram, internal clock divided by 1****Figure 154. Counter timing diagram, internal clock divided by 2**

Figure 155. Counter timing diagram, internal clock divided by 4

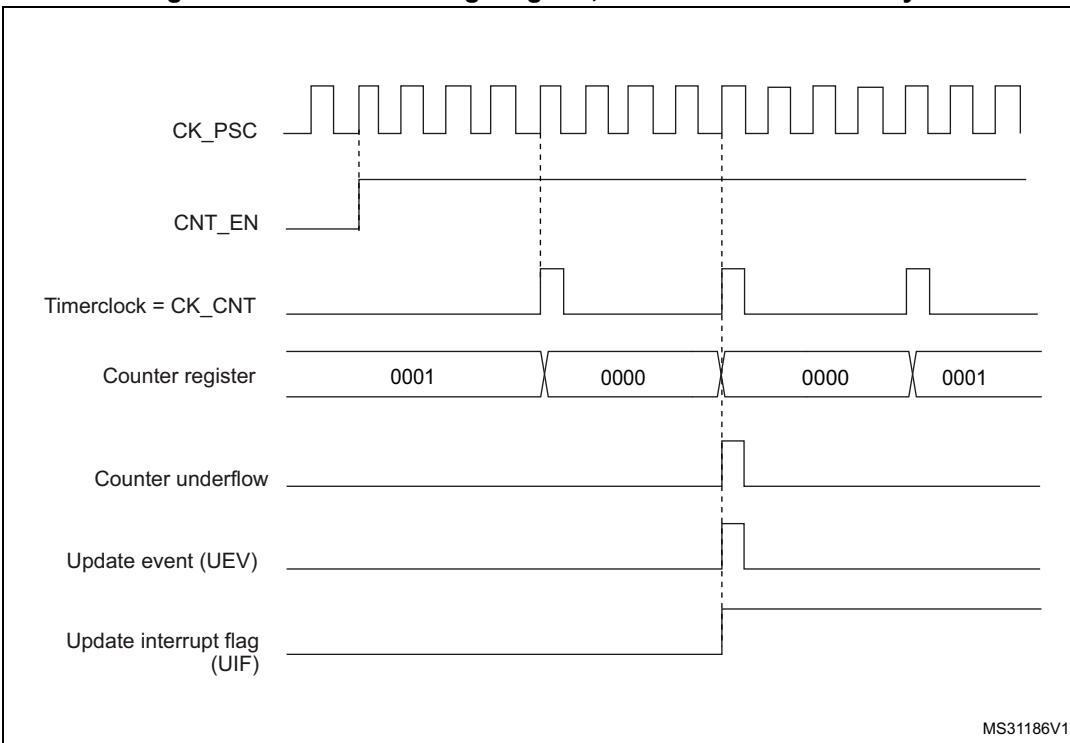
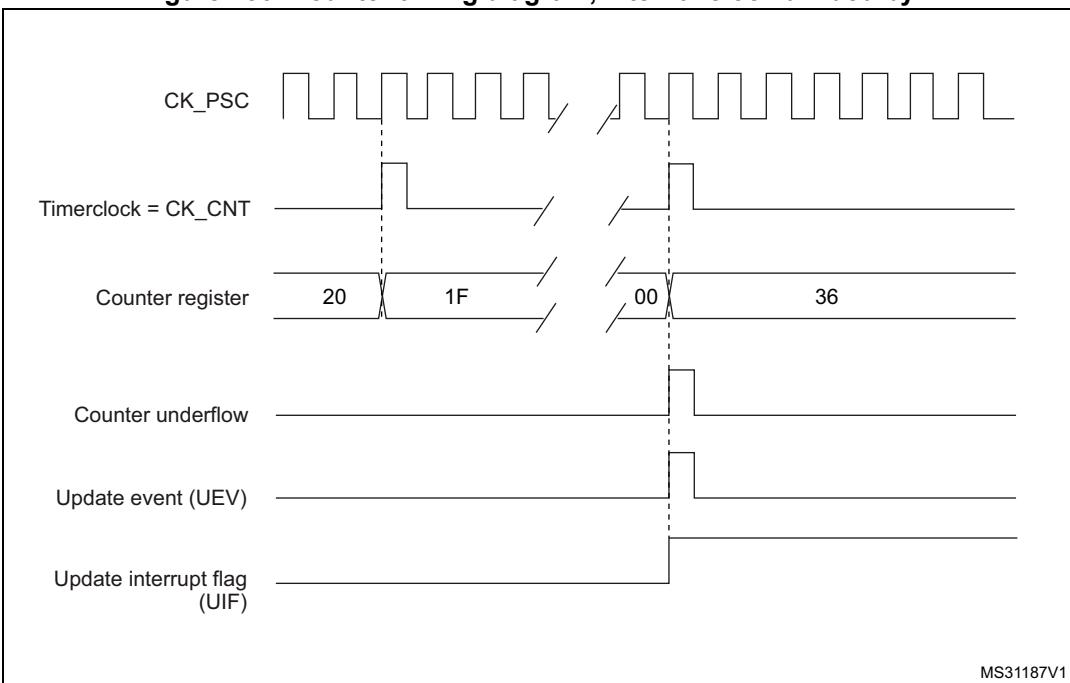
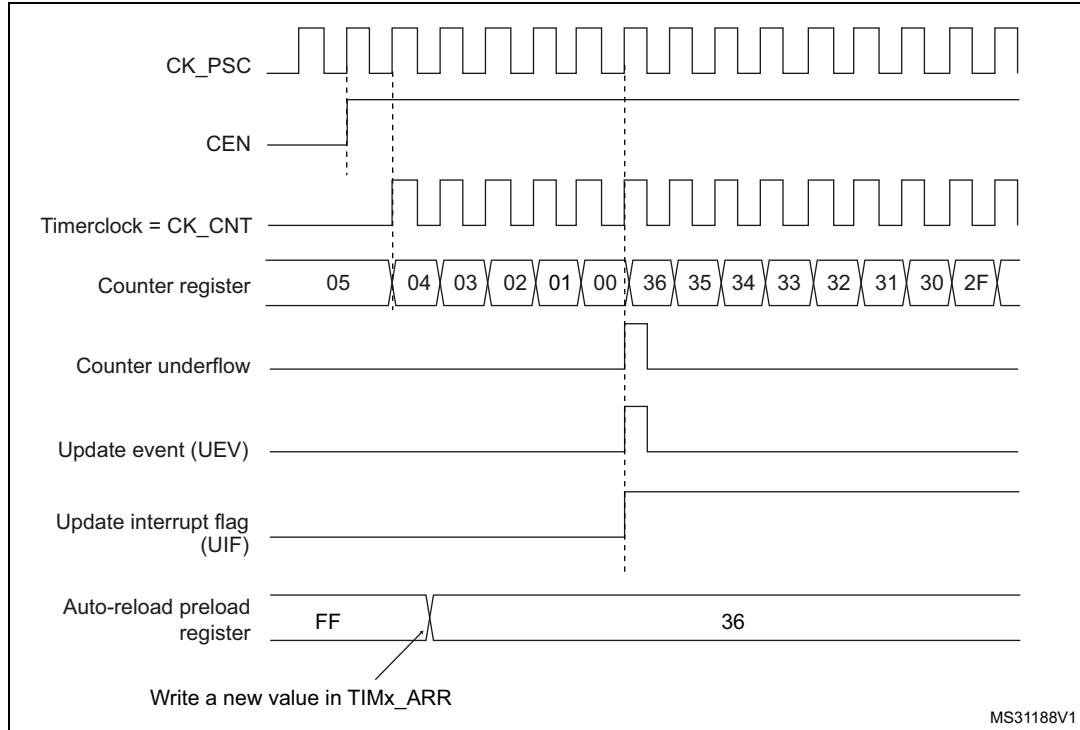


Figure 156. Counter timing diagram, internal clock divided by N



**Figure 157. Counter timing diagram, update event when repetition counter is not used**

### Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx\_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or

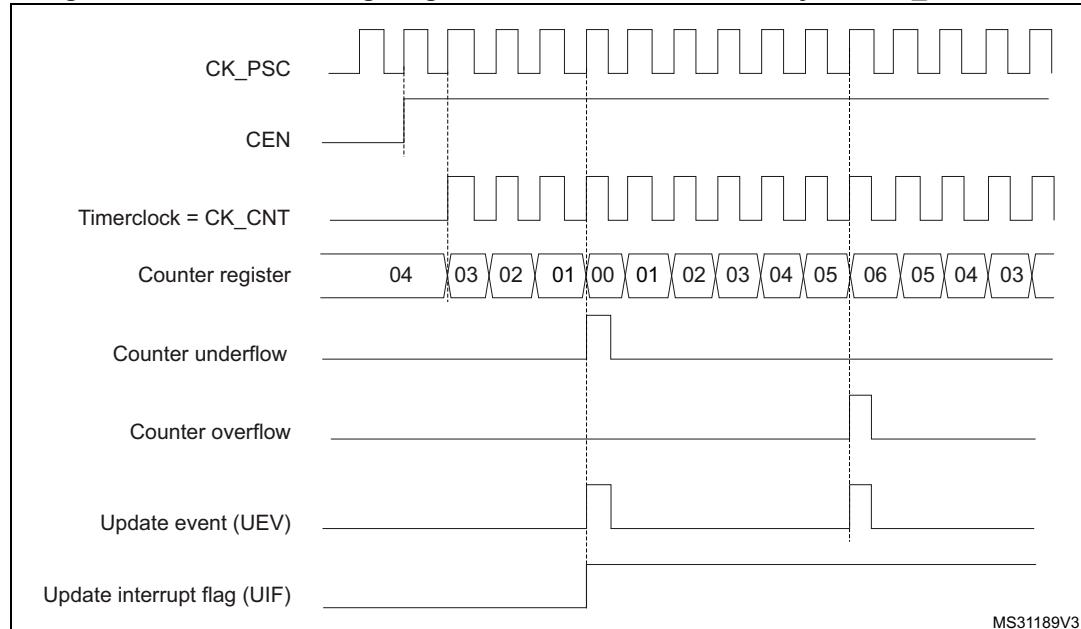
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx\_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

**Figure 158. 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 24.4: TIM1 registers](#)).

Figure 159. Counter timing diagram, internal clock divided by 2

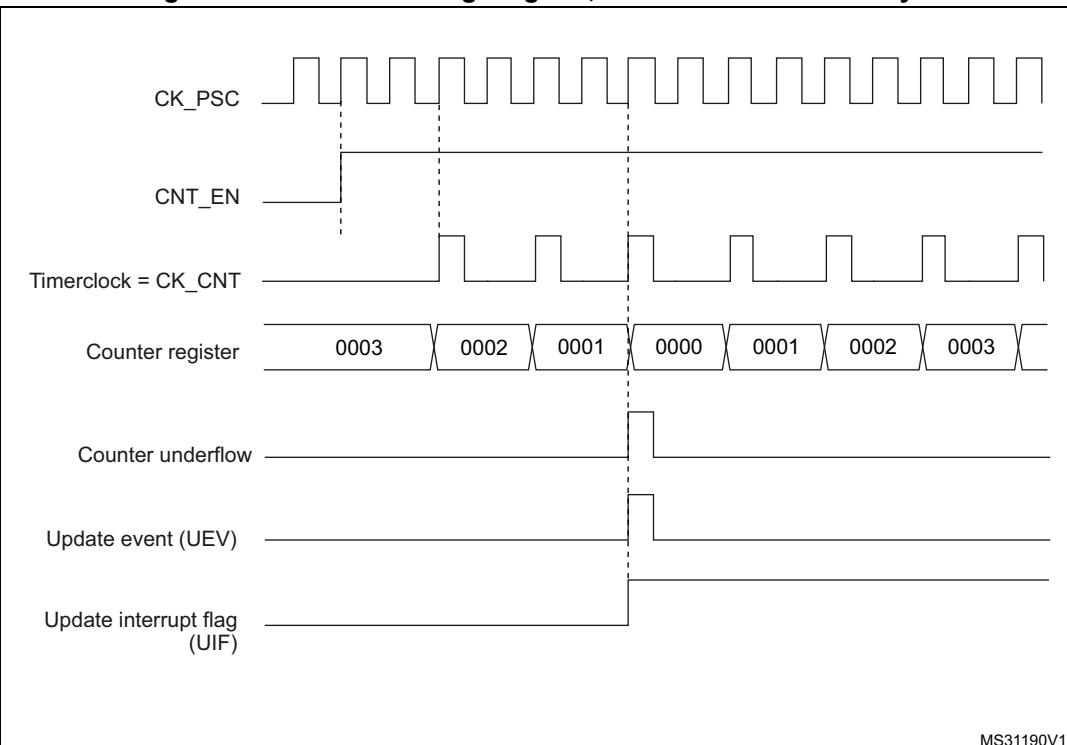


Figure 160. Counter timing diagram, internal clock divided by 4, TIMx\_ARR=0x36

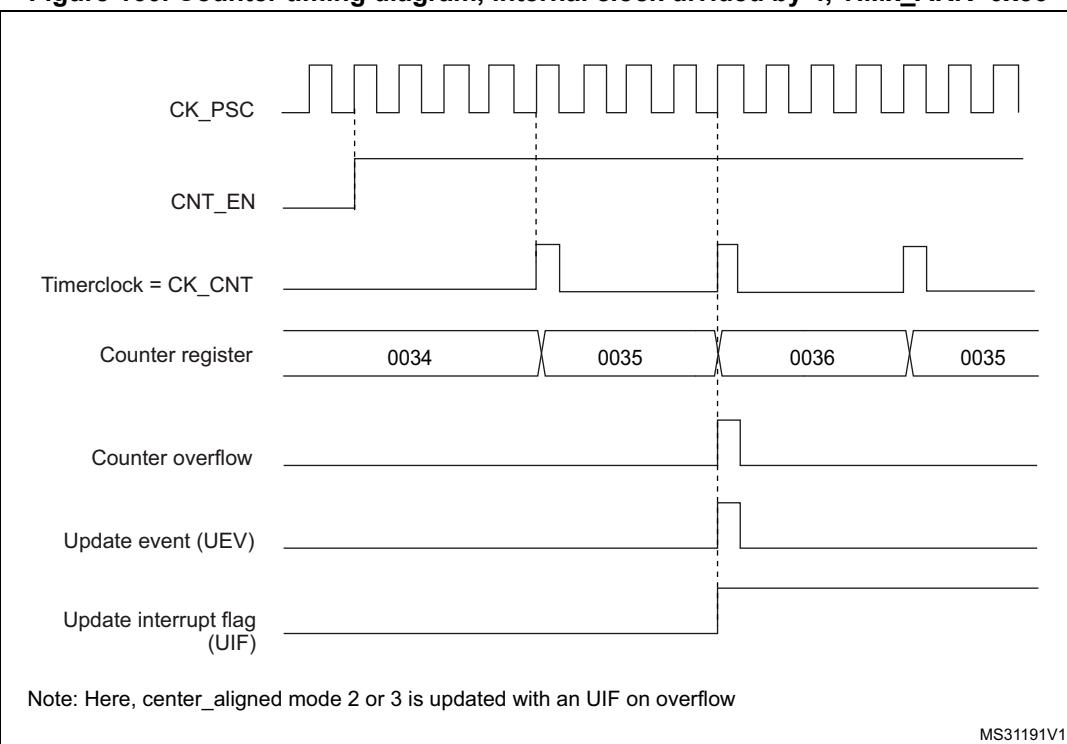


Figure 161. Counter timing diagram, internal clock divided by N

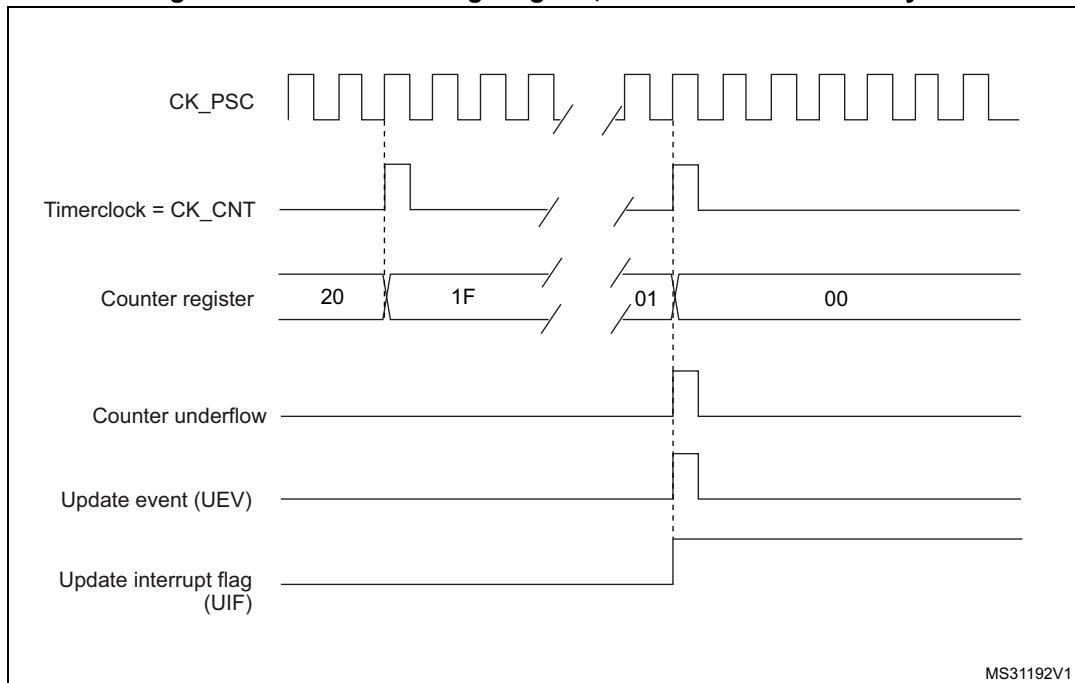
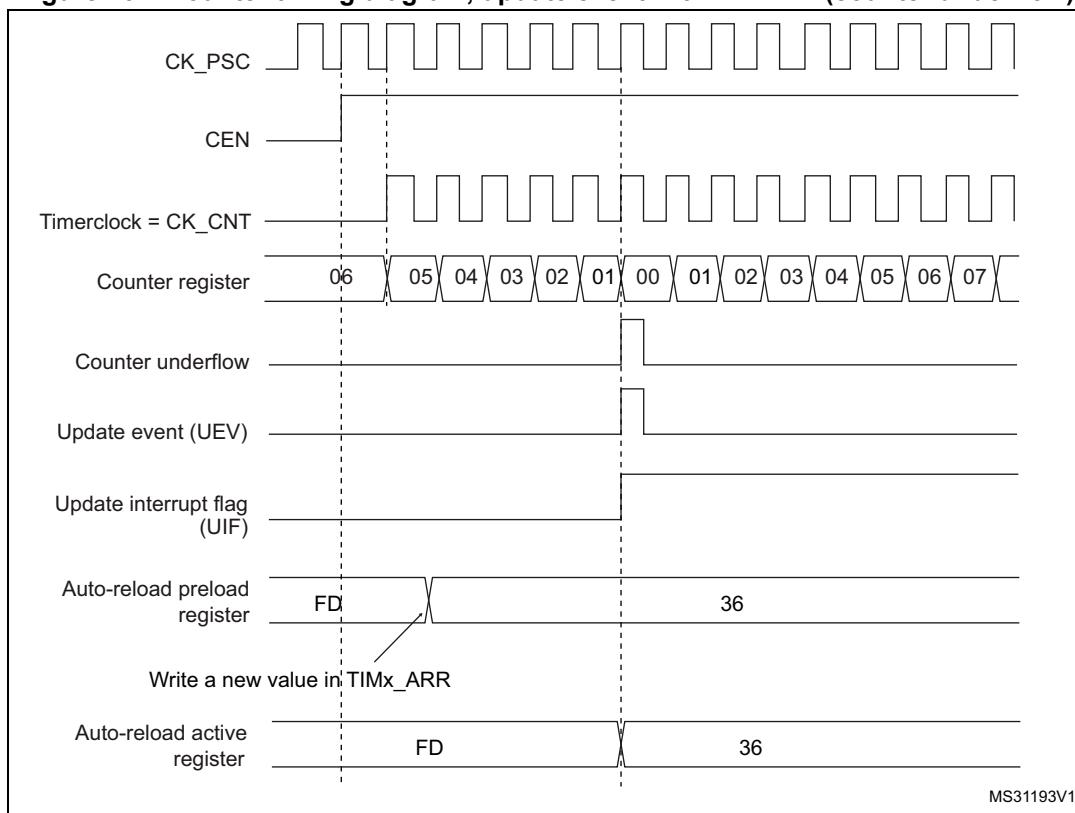
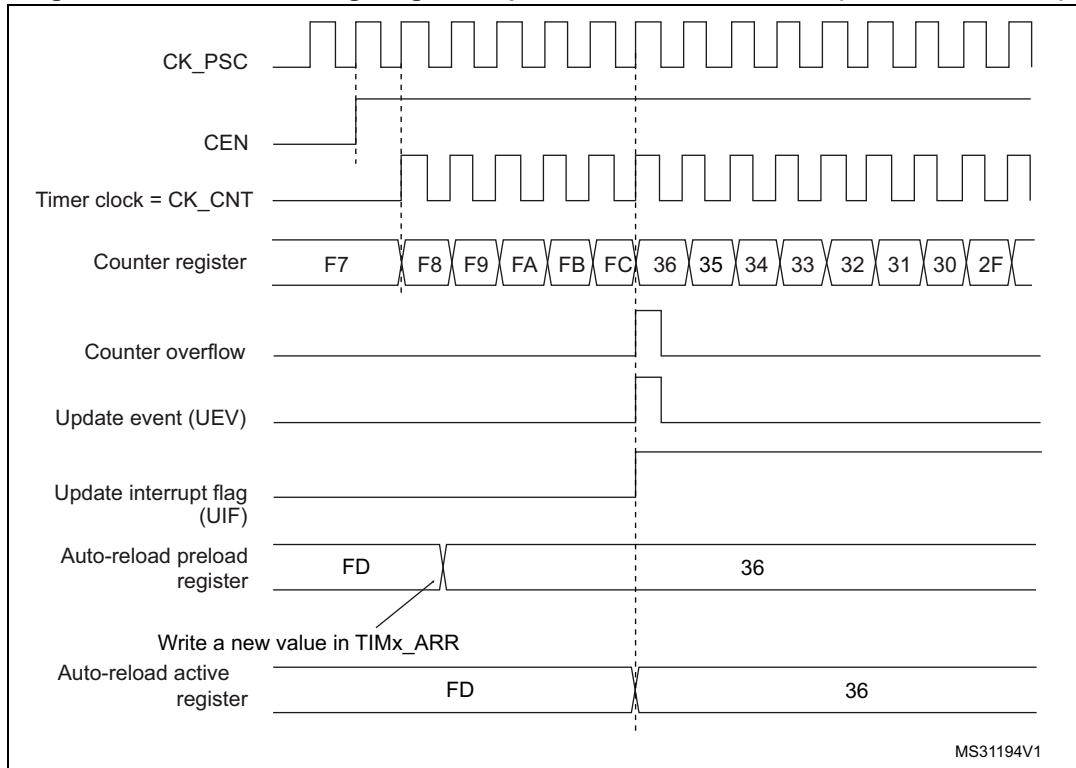


Figure 162. Counter timing diagram, update event with ARPE=1 (counter underflow)



**Figure 163. Counter timing diagram, Update event with ARPE=1 (counter overflow)**

### 24.3.3 Repetition counter

[Section 24.3.1: Time-base unit](#) describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx\_ARR auto-reload register, TIMx\_PSC prescaler register, but also TIMx\_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx\_RCR repetition counter register.

The repetition counter is decremented:

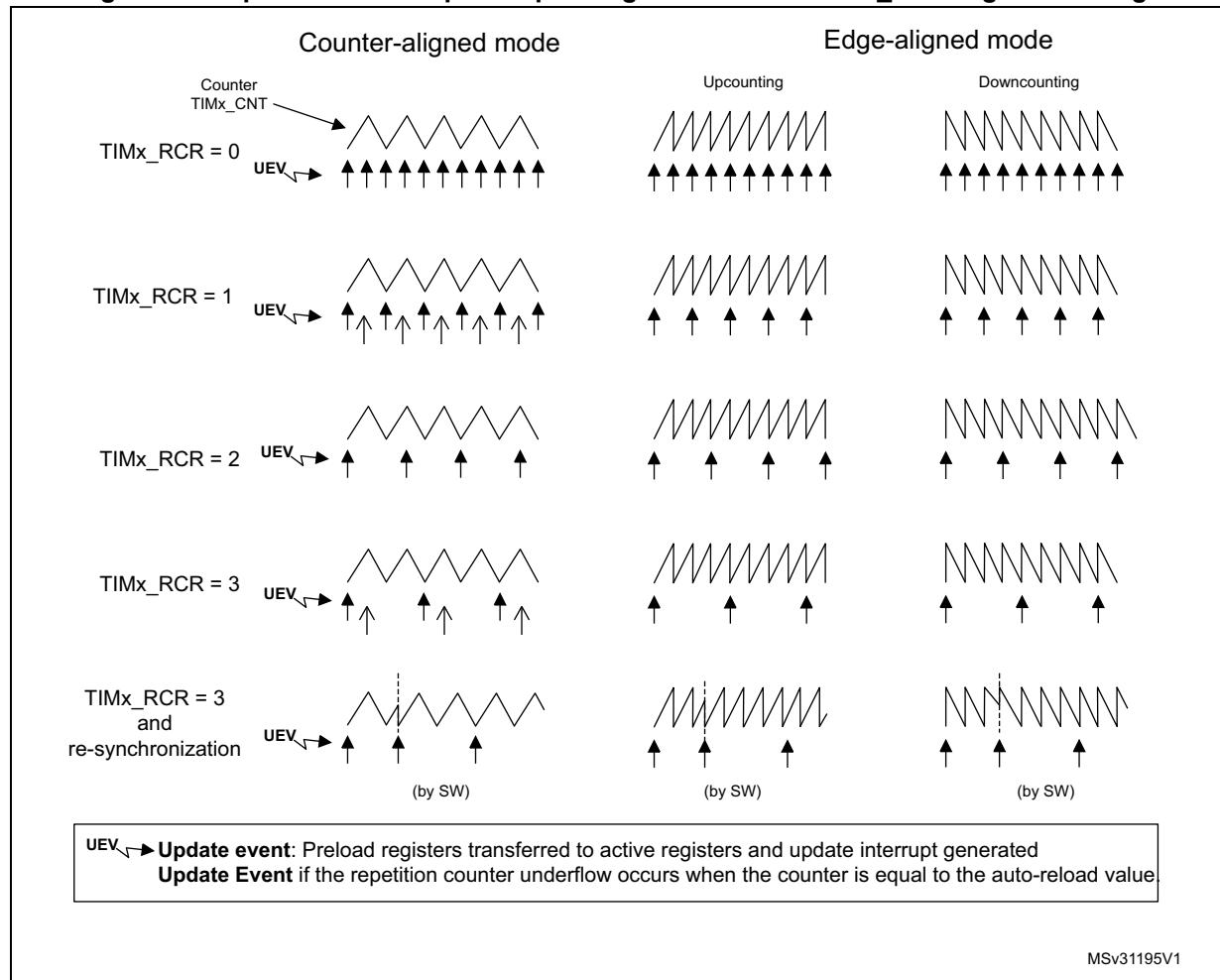
- At each counter overflow in upcounting mode,
  - At each counter underflow in downcounting mode,
  - At each counter overflow and at each counter underflow in center-aligned mode.
- Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is  $2 \times T_{ck}$ , due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx\_RCR register value (refer to [Figure 164](#)). When the update event is generated by software (by setting the UG bit in TIMx\_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx\_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

**Figure 164. Update rate examples depending on mode and TIMx\_RCR register settings**



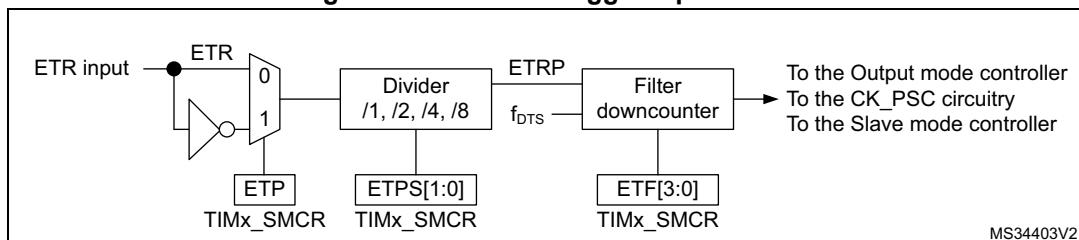
#### 24.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see [Section 24.3.5](#))
- trigger for the slave mode (see [Section 24.3.26](#))
- PWM reset input for cycle-by-cycle current regulation (see [Section 24.3.7](#))

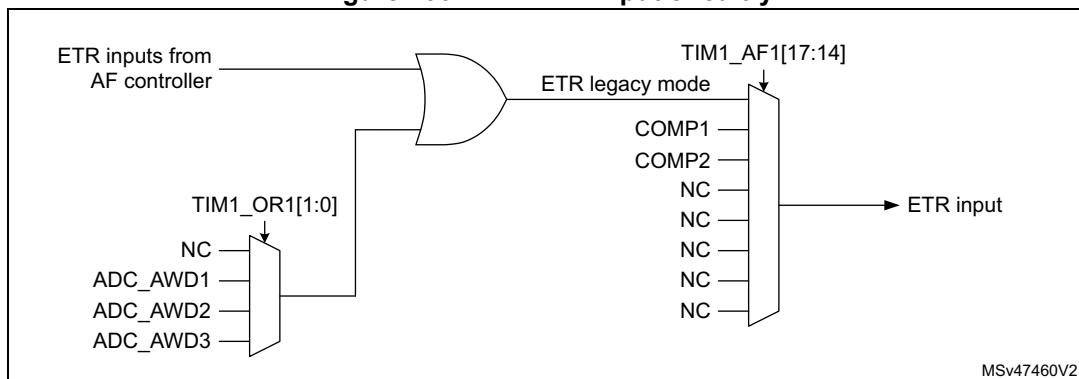
[Figure 165](#) below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

**Figure 165. External trigger input block**



The ETR input comes from multiple sources: input pins (default configuration), comparator outputs and analog watchdogs. The selection is done with the ETRSEL[3:0] and the TIM1\_OR1[1:0] bitfields.

**Figure 166. TIM1 ETR input circuitry**



### 24.3.5 Clock selection

The counter clock can be provided by the following clock sources:

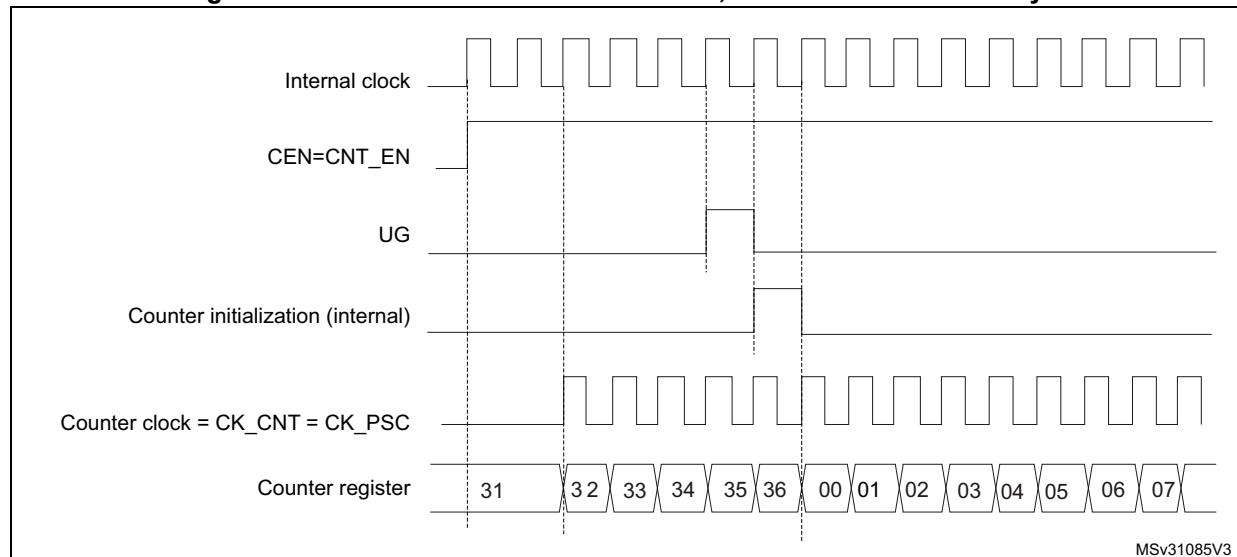
- Internal clock (CK\_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

#### Internal clock source (CK\_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK\_INT.

*Figure 167* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

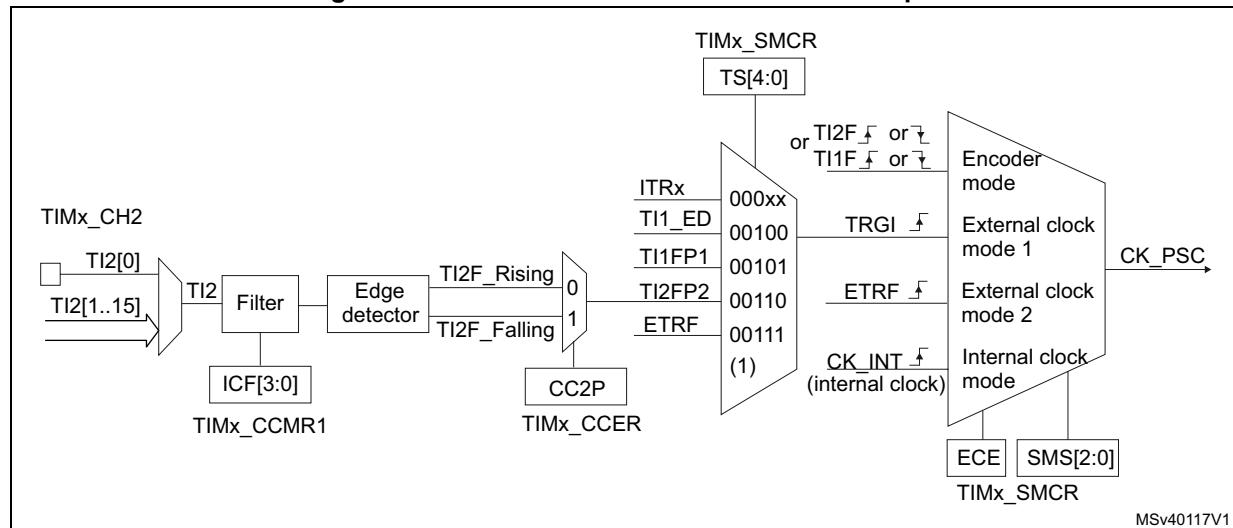
**Figure 167. Control circuit in normal mode, internal clock divided by 1**



#### 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 168. TI2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

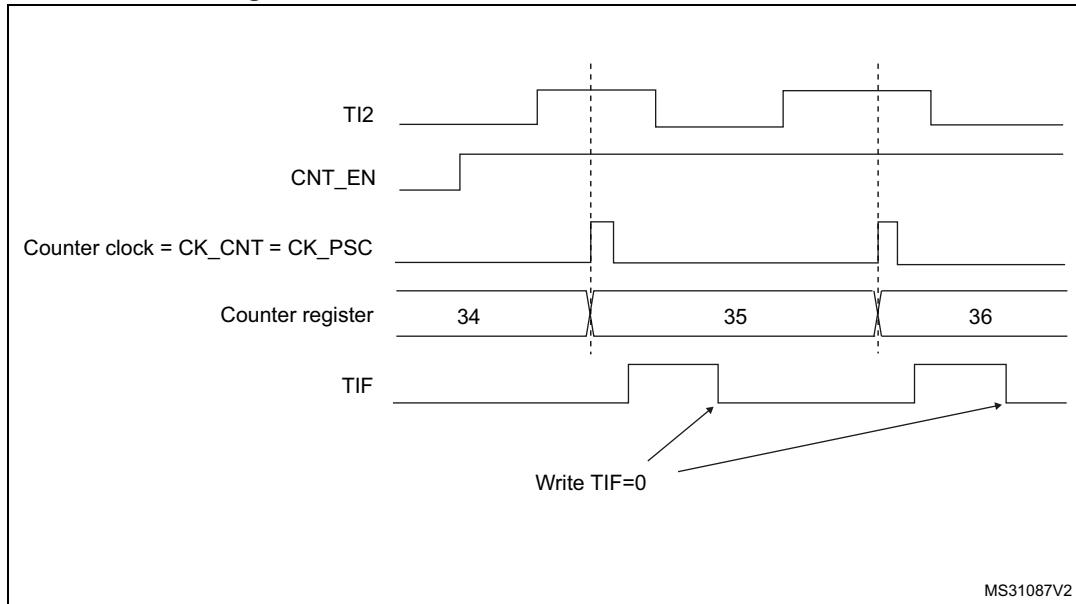
1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx\_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx\_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx\_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx\_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

*Note:* The capture prescaler is not used for triggering, so the user does not need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

**Figure 169. Control circuit in external clock mode 1**



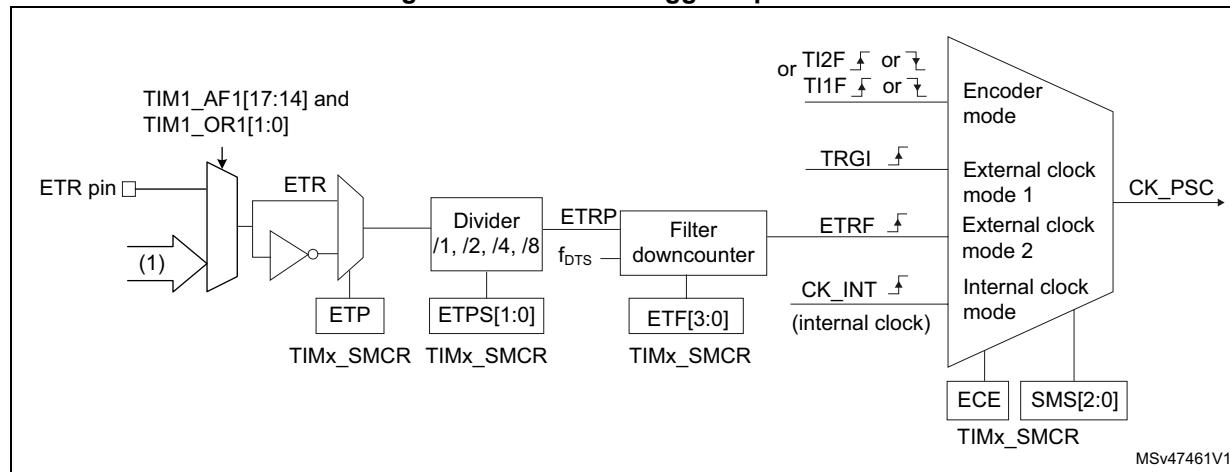
## External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx\_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The [Figure 170](#) gives an overview of the external trigger input block.

**Figure 170. External trigger input block**



1. Refer to *Figure 166: TIM1 ETR input circuitry*.

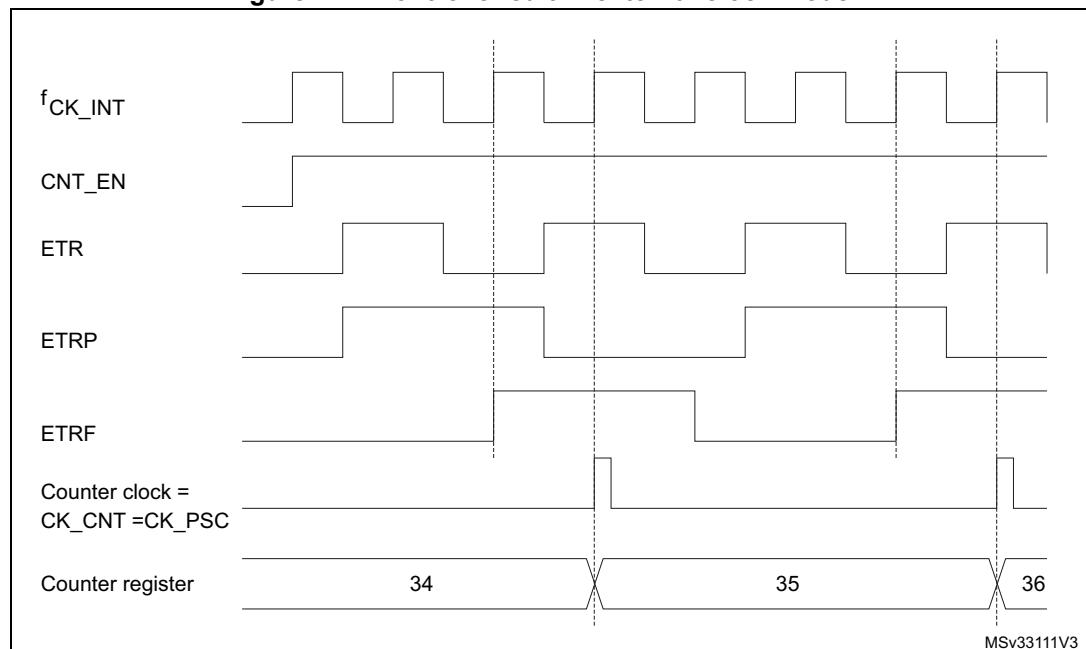
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx\_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx\_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx\_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx\_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most  $\frac{1}{4}$  of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by proper ETPS prescaler setting.

**Figure 171. Control circuit in external clock mode 2**



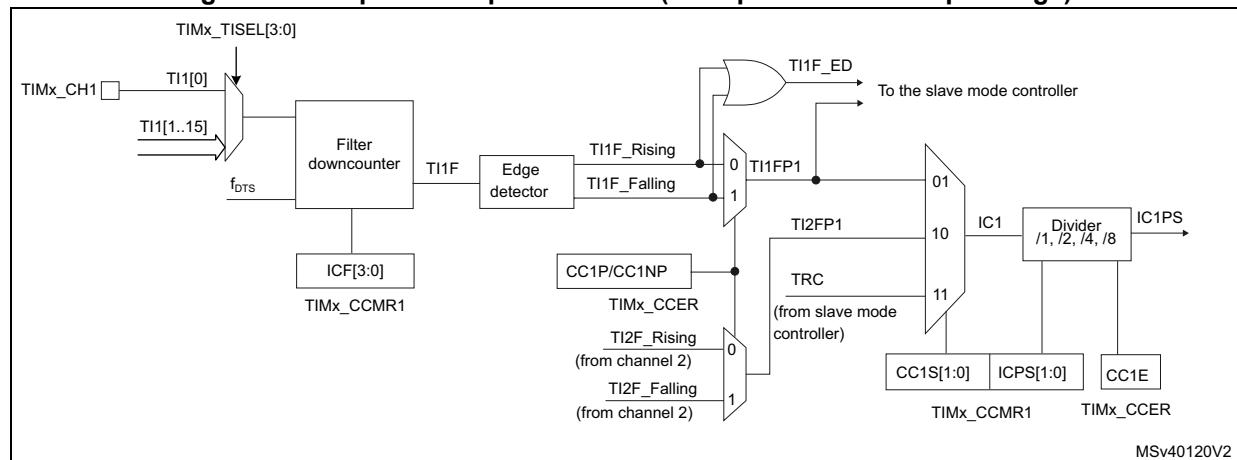
### 24.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

*Figure 172 to Figure 175* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TI<sub>x</sub> input to generate a filtered signal TI<sub>x</sub>F. Then, an edge detector with polarity selection generates a signal (TI<sub>x</sub>FP<sub>x</sub>) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (IC<sub>x</sub>PS).

**Figure 172. Capture/compare channel (example: channel 1 input stage)**



The output stage generates an intermediate waveform which is then used for reference: OC<sub>x</sub>Ref (active high). The polarity acts at the end of the chain.

**Figure 173. Capture/compare channel 1 main circuit**

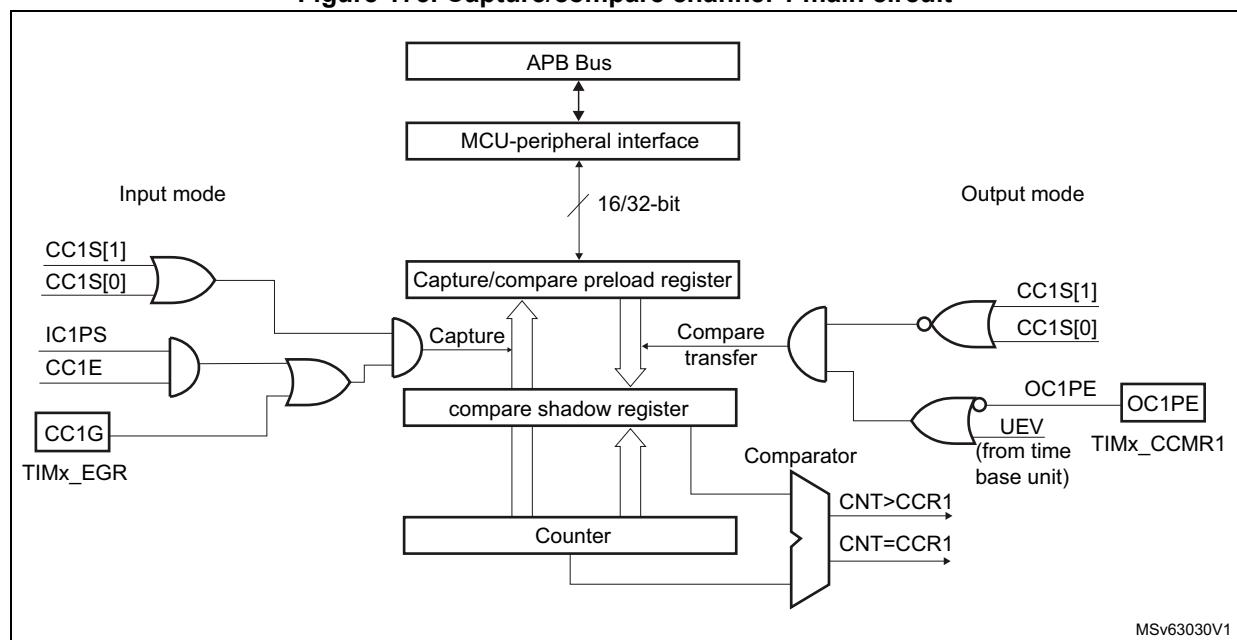
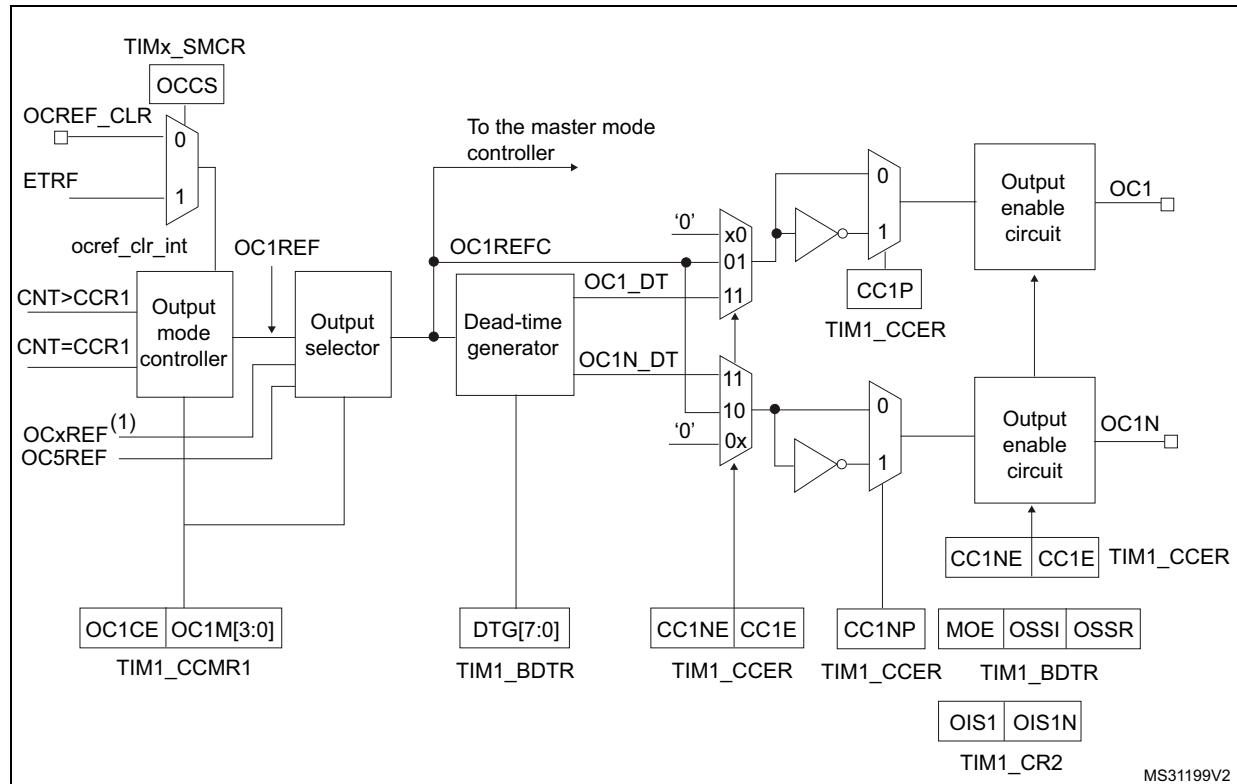


Figure 174. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)



1. OCxREF, where x is the rank of the complementary channel

Figure 175. Output stage of capture/compare channel (channel 4)

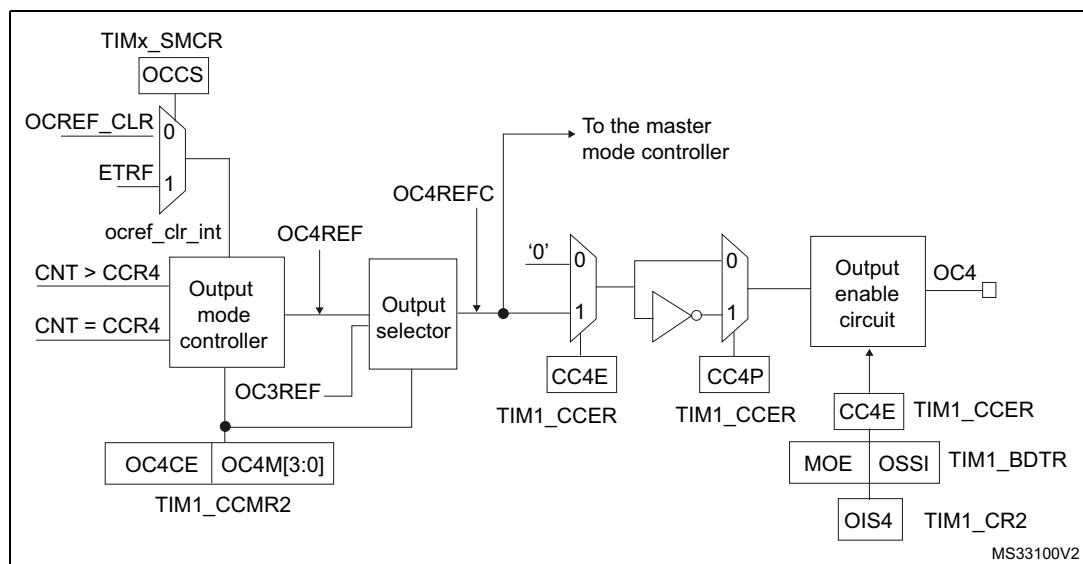
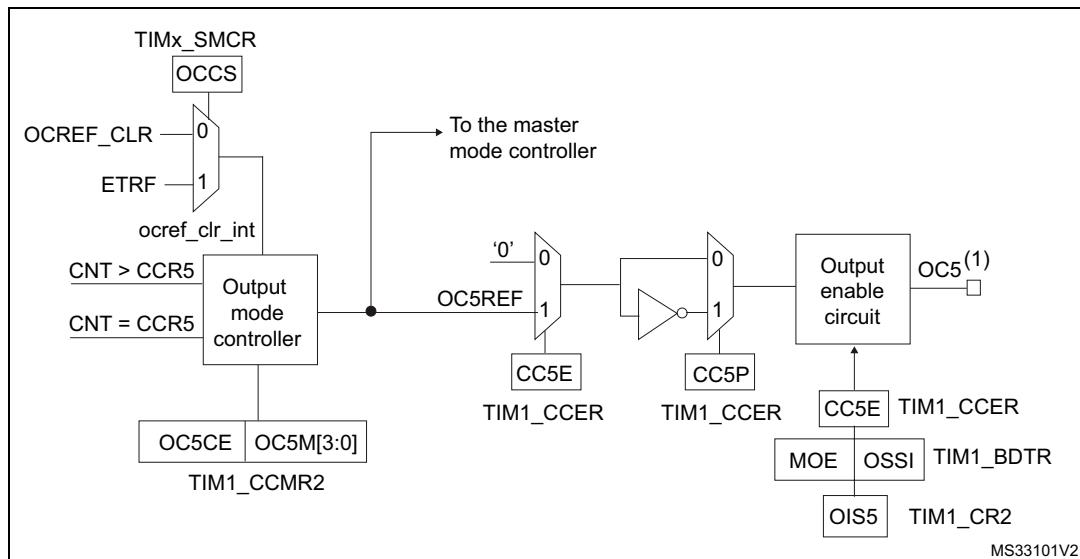


Figure 176. Output stage of capture/compare channel (channel 5, idem ch. 6)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

#### 24.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx\_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx\_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx\_CCRx register. CCxOF is cleared when written with '0'.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx\_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx\_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at  $f_{DTS}$  frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx\_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx\_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx\_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

*Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx\_EGR register.*

#### 24.3.8 PWM input mode

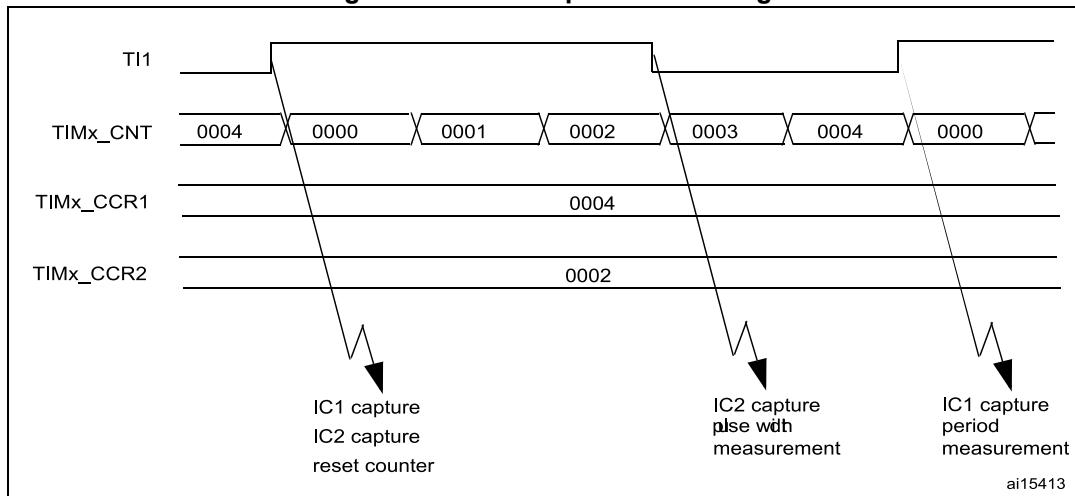
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, the user can measure the period (in TIMx\_CCR1 register) and the duty cycle (in TIMx\_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK\_INT frequency and prescaler value):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input for TIMx\_CCR1: write the CC1S bits to 01 in the TIMx\_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx\_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
4. Select the active input for TIMx\_CCR2: write the CC2S bits to 10 in the TIMx\_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx\_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP='10' (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx\_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx\_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx\_CCER register.

Figure 177. PWM input mode timing



### 24.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 0100 in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 24.3.10 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while Channel 5 and 6 are only available inside the device (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCxM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx\_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register.

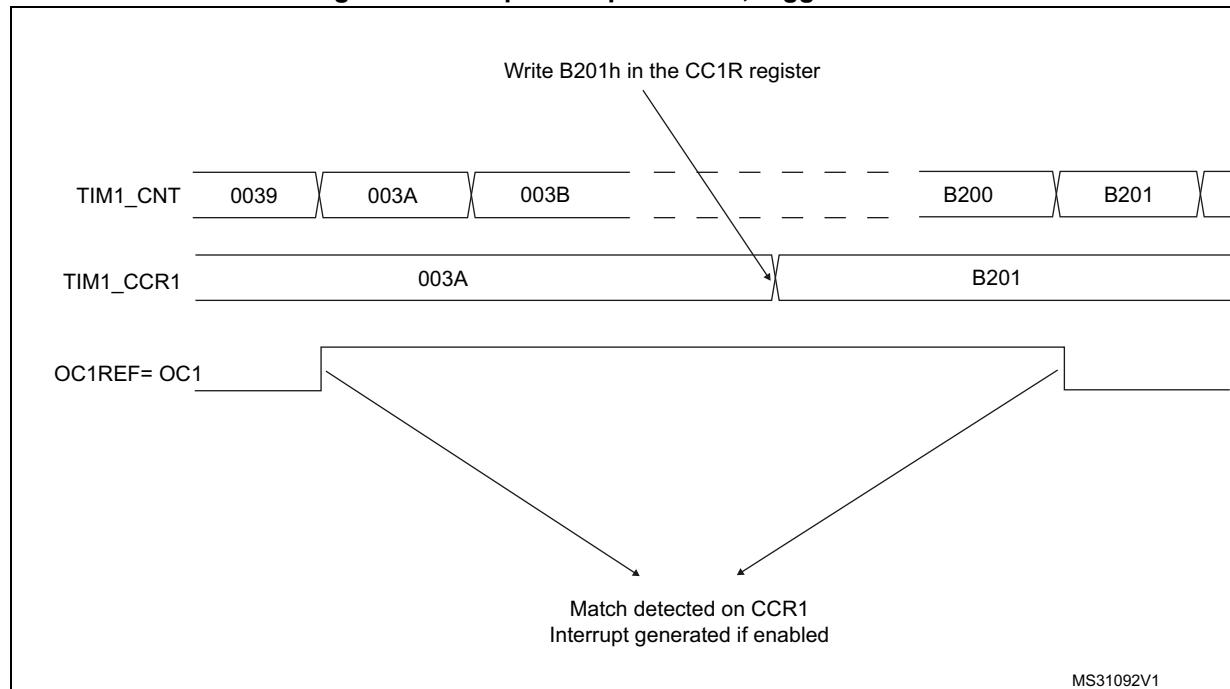
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

#### Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
  - Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
  - Write OCxPE = 0 to disable preload register
  - Write CCxP = 0 to select active high polarity
  - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 178](#).

Figure 178. Output compare mode, toggle on OC1



#### 24.3.11 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '0110' (PWM mode 1) or '0111' (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSS1 and OSSR bits (TIMx\_CCER and TIMx\_BDTR registers). Refer to the TIMx\_CCER register description for more details.

In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether TIMx\_CCRx  $\leq$  TIMx\_CNT or TIMx\_CNT  $\leq$  TIMx\_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx\_CR1 register.

### PWM edge-aligned mode

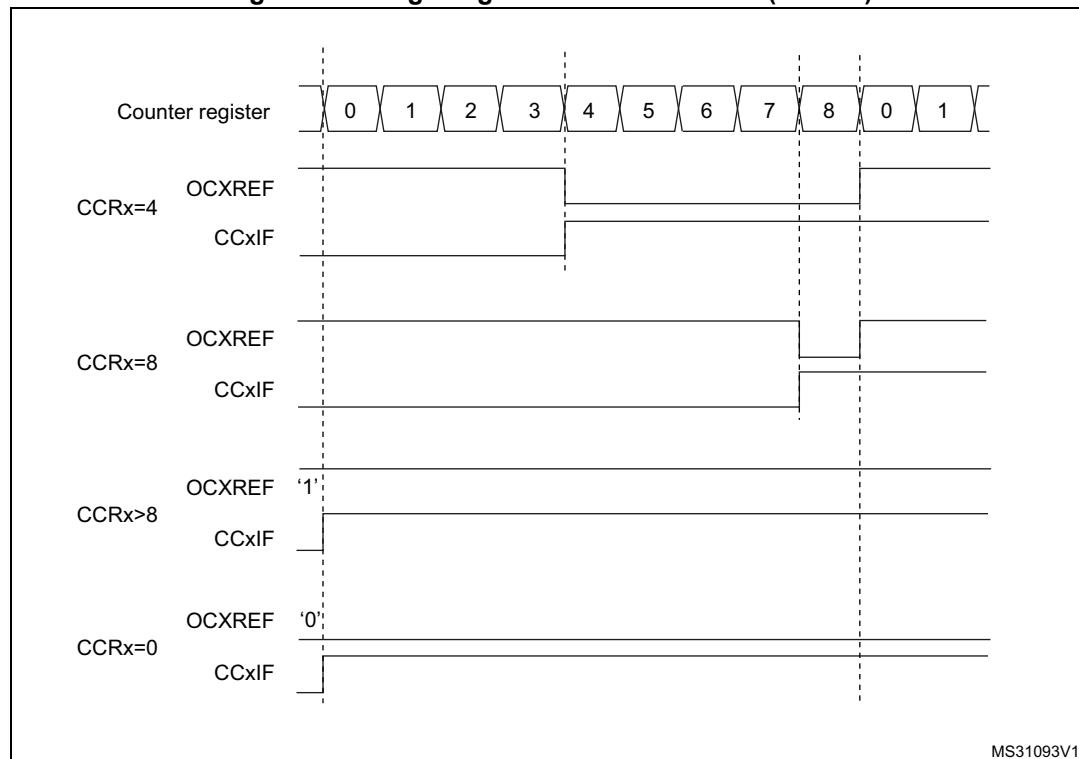
- Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low. Refer to the [Upcounting mode on page 666](#).

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as  $\text{TIMx\_CNT} < \text{TIMx\_CCR}_x$  else it becomes low. If the compare value in  $\text{TIMx\_CCR}_x$  is greater than the auto-reload value (in  $\text{TIMx\_ARR}$ ) then OCxREF is held at '1'. If the compare value is 0 then OCxRef is held at '0'.

[Figure 179](#) shows some edge-aligned PWM waveforms in an example where  $\text{TIMx\_ARR}=8$ .

**Figure 179. Edge-aligned PWM waveforms (ARR=8)**



- Downcounting configuration

Downcounting is active when DIR bit in TIMx\_CR1 register is high. Refer to the [Downcounting mode on page 670](#)

In PWM mode 1, the reference signal OCxRef is low as long as  $\text{TIMx\_CNT} > \text{TIMx\_CCR}_x$  else it becomes high. If the compare value in  $\text{TIMx\_CCR}_x$  is greater than the auto-reload value in  $\text{TIMx\_ARR}$ , then OCxREF is held at '1'. 0% PWM is not possible in this mode.

### PWM center-aligned mode

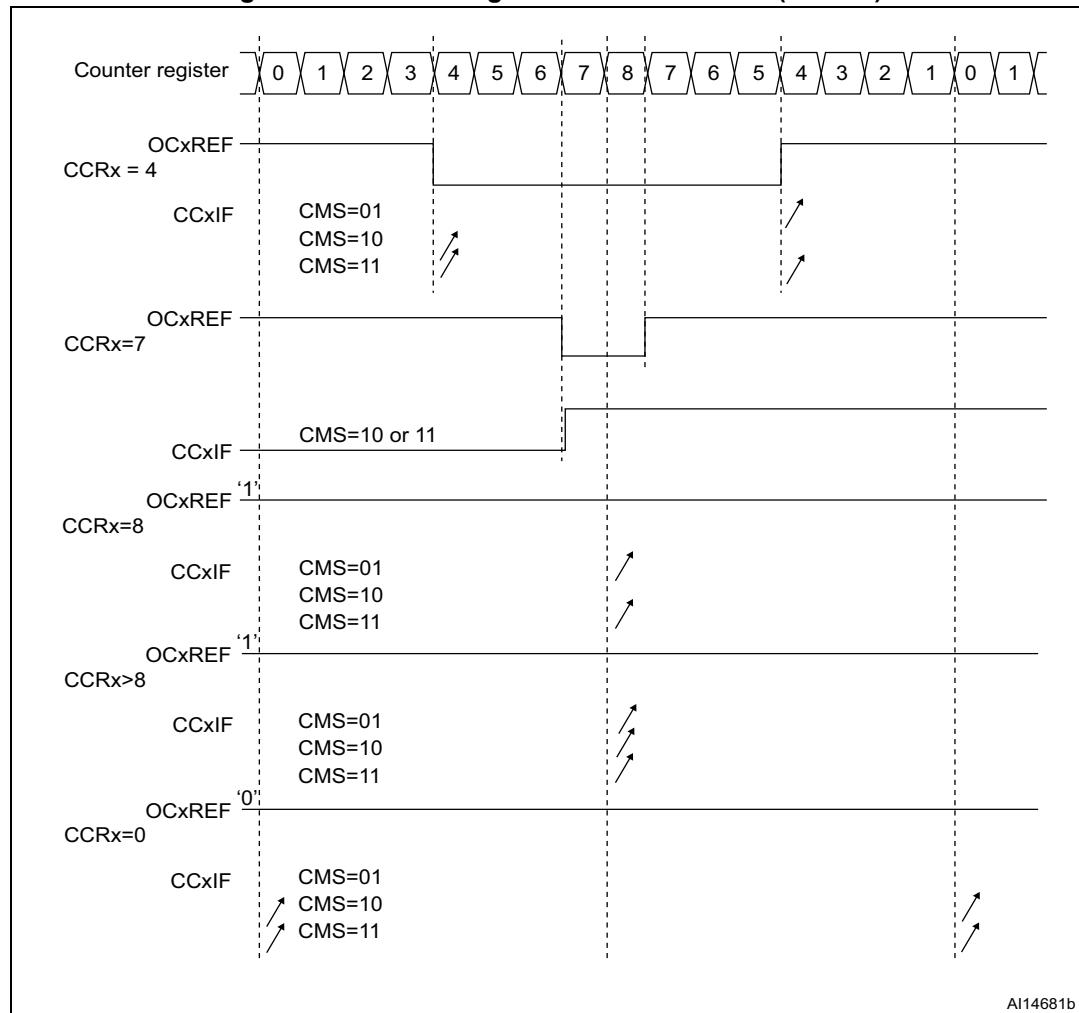
Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIMx\_CR1 register is updated by hardware and must not be changed by software. Refer to the [Center-aligned mode \(up/down counting\) on page 673](#).

[Figure 180](#) 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 180. Center-aligned PWM waveforms (ARR=8)**



AI14681b

#### Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

- in the TIMx\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
    - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx\_CNT>TIMx\_ARR). For example, if the counter was counting up, it continues to count up.
    - The direction is updated if 0 or the TIMx\_ARR value is written in the counter but no Update Event UEV is generated.
  - The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx\_EGR register) just before starting the counter and not to write the counter while it is running.

#### 24.3.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx\_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx\_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx\_CCR1 and TIMx\_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx\_CCR3 and TIMx\_CCR4

Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing '1110' (Asymmetric PWM mode 1) or '1111' (Asymmetric PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

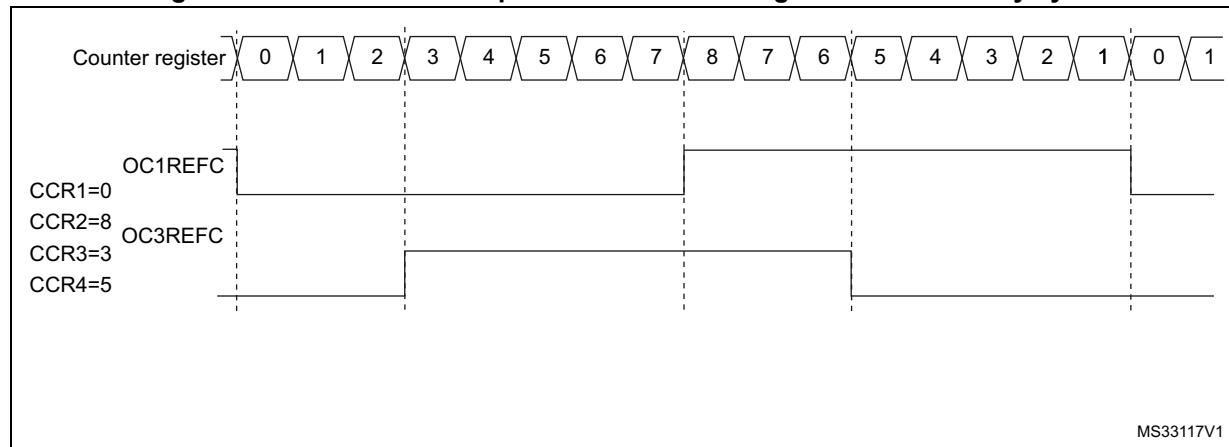
*Note:*

*The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.*

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 1.

*Figure 181* represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 181. Generation of 2 phase-shifted PWM signals with 50% duty cycle



### 24.3.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx\_ARR register, the duty cycle and delay are determined by the two TIMx\_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx\_CCR1 and TIMx\_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx\_CCR3 and TIMx\_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

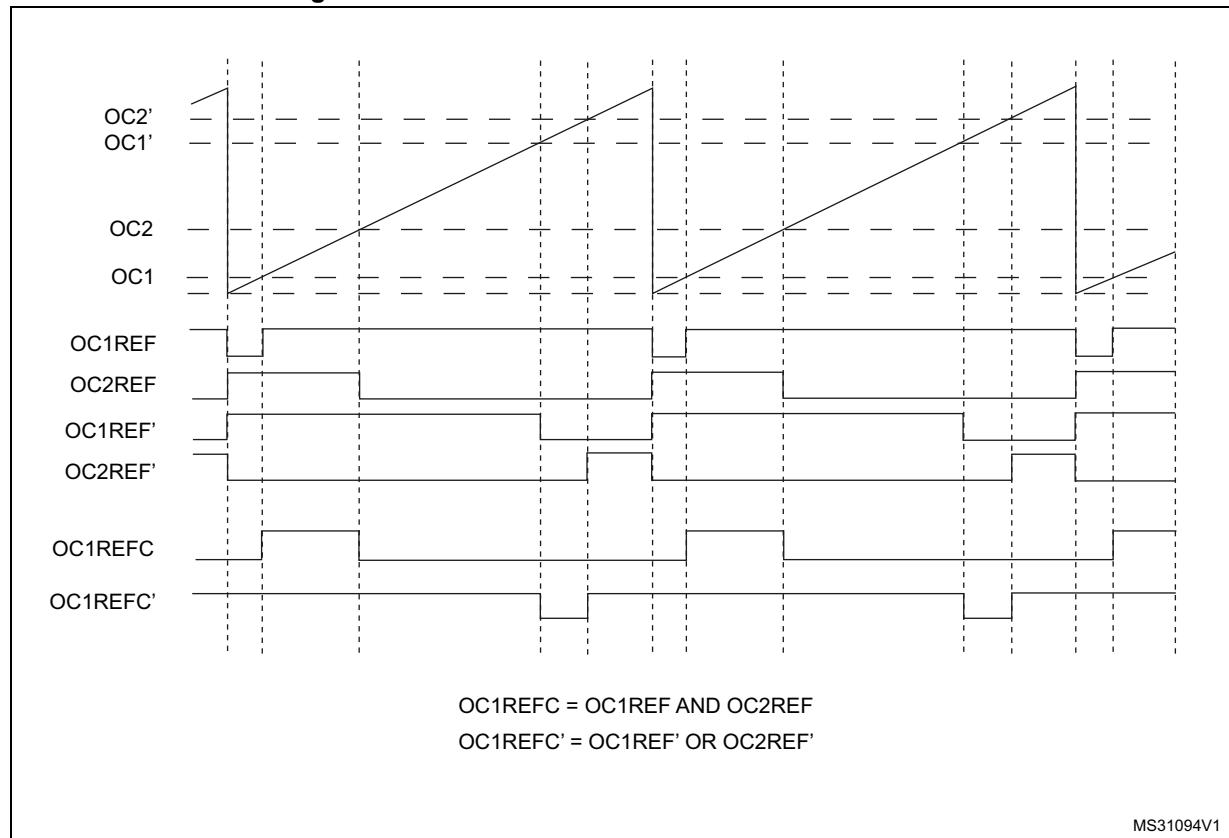
Note:

*The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.*

*Figure 182* represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1.

Figure 182. Combined PWM mode on channel 1 and 3



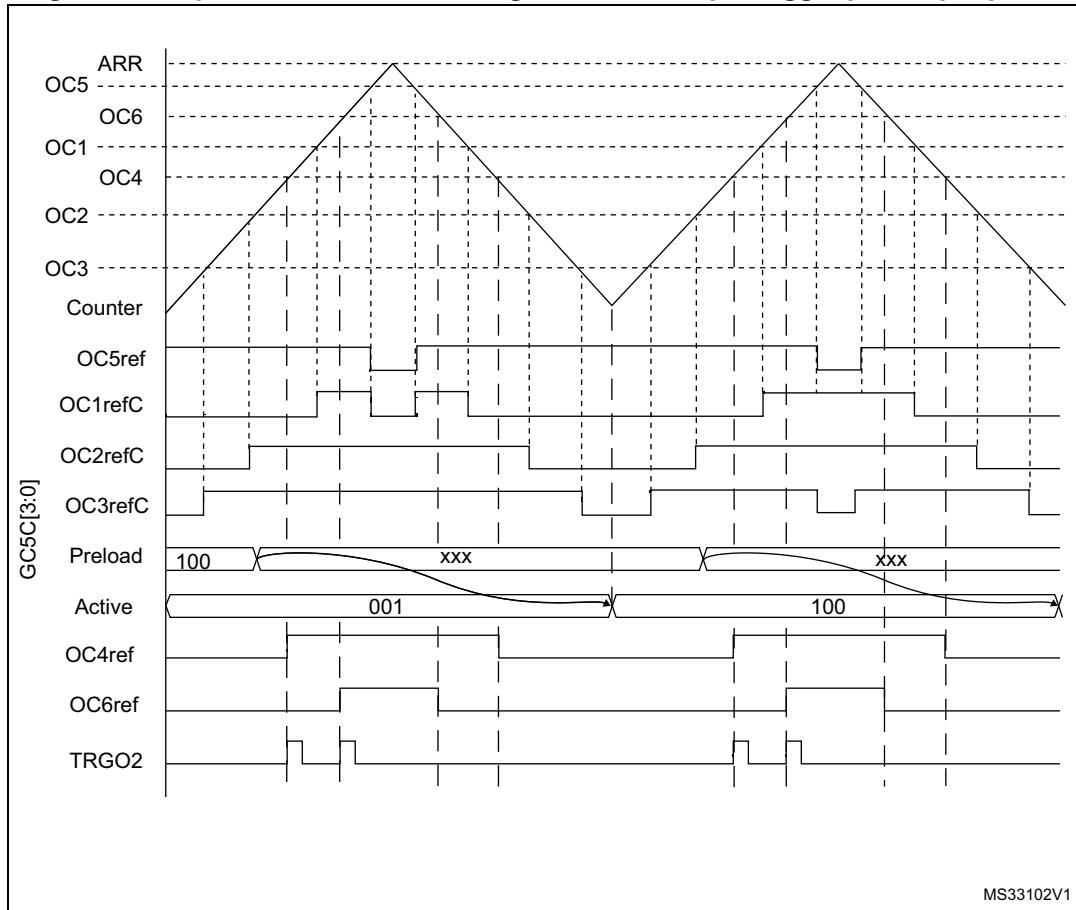
#### 24.3.14 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The OC5REF signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx\_CCR5 allow selection on which reference signal the OC5REF is combined. The resulting signals, OCxREFC, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, OC1REFC is controlled by TIMx\_CCR1 and TIMx\_CCR5
- If GC5C2 is set, OC2REFC is controlled by TIMx\_CCR2 and TIMx\_CCR5
- If GC5C3 is set, OC3REFC is controlled by TIMx\_CCR3 and TIMx\_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 183. 3-phase combined PWM signals with multiple trigger pulses per period



The TRGO2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to [Section 24.3.27: ADC synchronization](#) for more details.

#### 24.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OC<sub>x</sub> or complementary OC<sub>xN</sub>) can be selected independently for each output. This is done by writing to the CC<sub>xP</sub> and CC<sub>xNP</sub> bits in the TIM<sub>x</sub>\_CCER register.

The complementary signals OC<sub>x</sub> and OC<sub>xN</sub> are activated by a combination of several control bits: the CC<sub>xE</sub> and CC<sub>xNE</sub> bits in the TIM<sub>x</sub>\_CCER register and the MOE, OIS<sub>x</sub>, OIS<sub>xN</sub>, OSS<sub>I</sub> and OSS<sub>R</sub> bits in the TIM<sub>x</sub>\_BDTR and TIM<sub>x</sub>\_CR2 registers. Refer to

[Table 163: Output control bits for complementary OC<sub>x</sub> and OC<sub>xN</sub> channels with break feature on page 742](#) for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

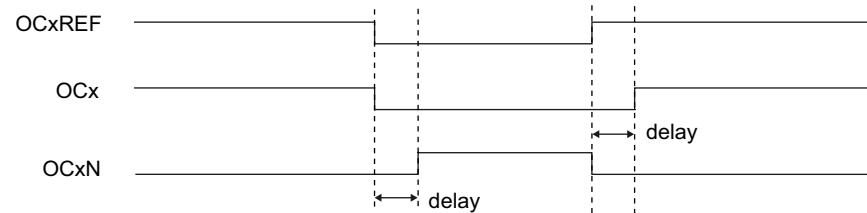
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

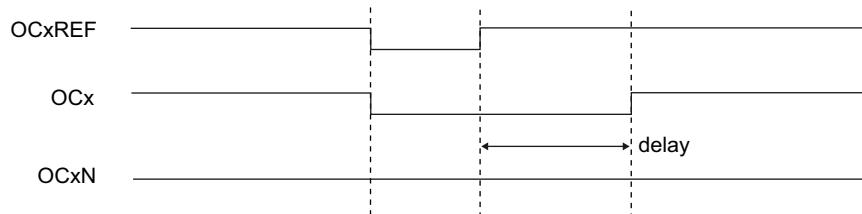
The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

**Figure 184. Complementary output with dead-time insertion**



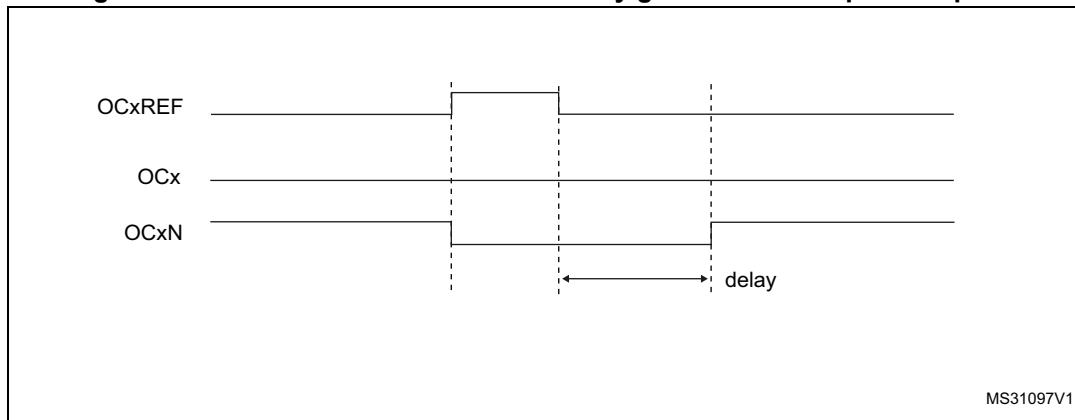
MS31095V1

**Figure 185. Dead-time waveforms with delay greater than the negative pulse**



MS31096V1

Figure 186. 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 24.4.20: TIM1 break and dead-time register \(TIM1\\_BDTR\)](#) for delay calculation.

#### Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx\_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

**Note:** When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

#### 24.3.16 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 timer. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx\_BDTR register allows the outputs to be enabled/disabled by software and is reset in case of break or break2 event.
- the OSS1 bit in the TIMx\_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx\_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to [Table 163: Output control bits for complementary OCx and OCxN channels with break feature on page 742](#) for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx\_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKE/BK2E and BKP/BK2P can be modified at the same time. When the BKE/BK2E and BKP/BK2P bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx\_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx\_AF1 and TIMx\_AF2 registers.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the SYSCFG\_CFGR2 register), with polarity selection and optional digital filtering
- An internal source:
  - the CPU1 Cortex®-M4 LOCKUP output
  - the PVD output
  - the SRAM parity error signal
  - a flash memory ECC double error detection
  - a clock failure event generated by the CSS detector
  - the output from a comparator, with polarity selection and optional digital filtering

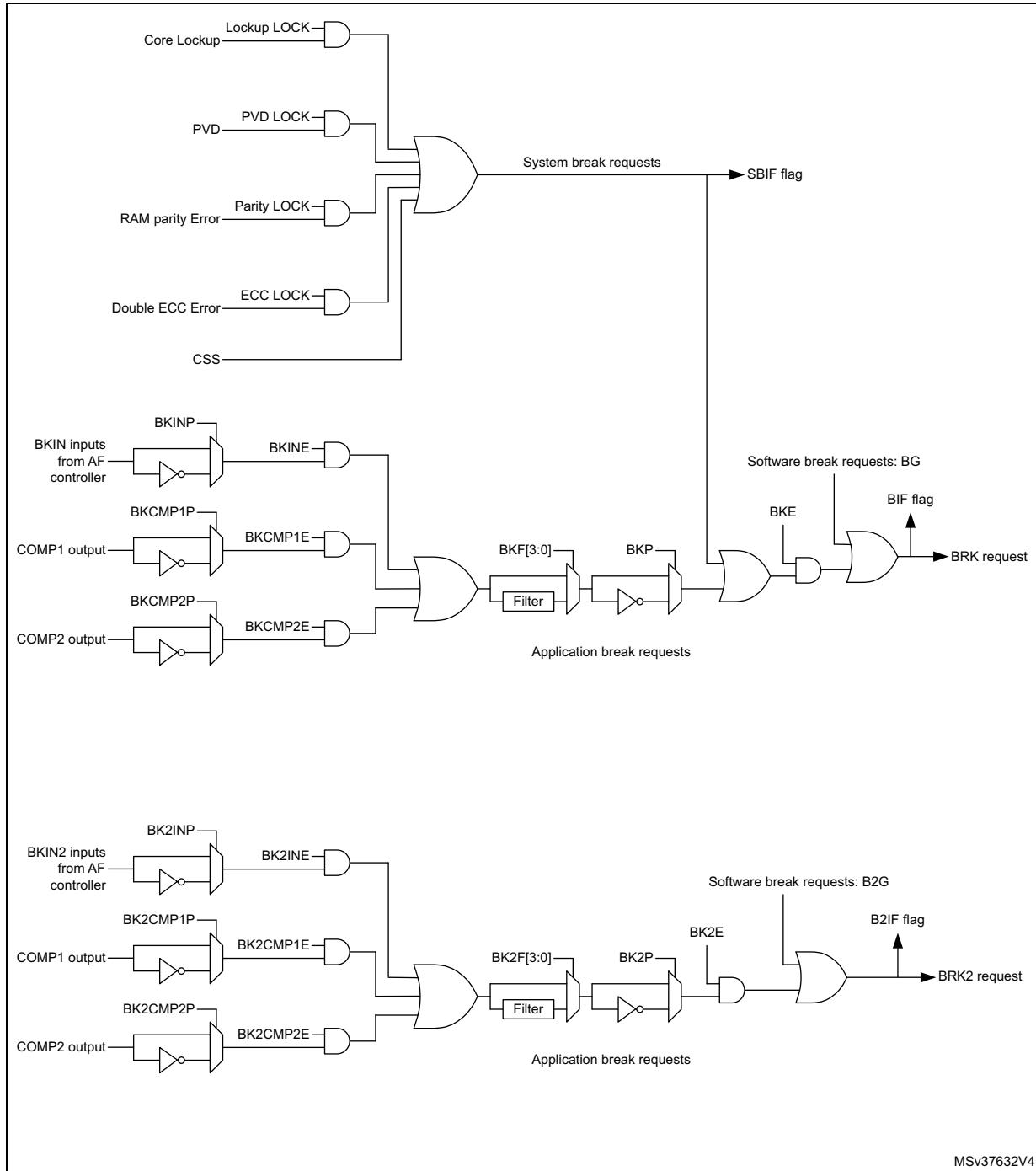
The sources for break2 (BRK2) are:

- An external source connected to one of the BKIN pin (as per selection done in the SYSCFG\_CFGR2 register), with polarity selection and optional digital filtering
- An internal source coming from a comparator output.

Break events can also be generated by software using BG and B2G bits in the TIMx\_EGR register. The software break generation using BG and B2G is active whatever the BKE and BK2E enable bits values.

All sources are ORed before entering the timer BRK or BRK2 inputs, as per [Figure 187](#) below.

**Figure 187. Break and Break2 circuitry overview**



**Note:** An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSS1 bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx\_CR2 register as soon as MOE=0. If OSS1=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 ck\_tim clock cycles).
  - If OSS1=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (SBIF, BIF and B2IF bits in the TIMx\_SR register) is set. An interrupt is generated if the BIE bit in the TIMx\_DIER register is set.
- If the AOE bit in the TIMx\_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note:

*If the MOE is reset by the CPU while the AOE bit is set, the outputs will be in idle state and forced to inactive level or Hi-Z depending on OSS1 value.*

*If both the MOE and AOE bits are reset by the CPU, the outputs will be in disabled state and driven with the level programmed in the OISx bit in the TIMx\_CR2 register.*

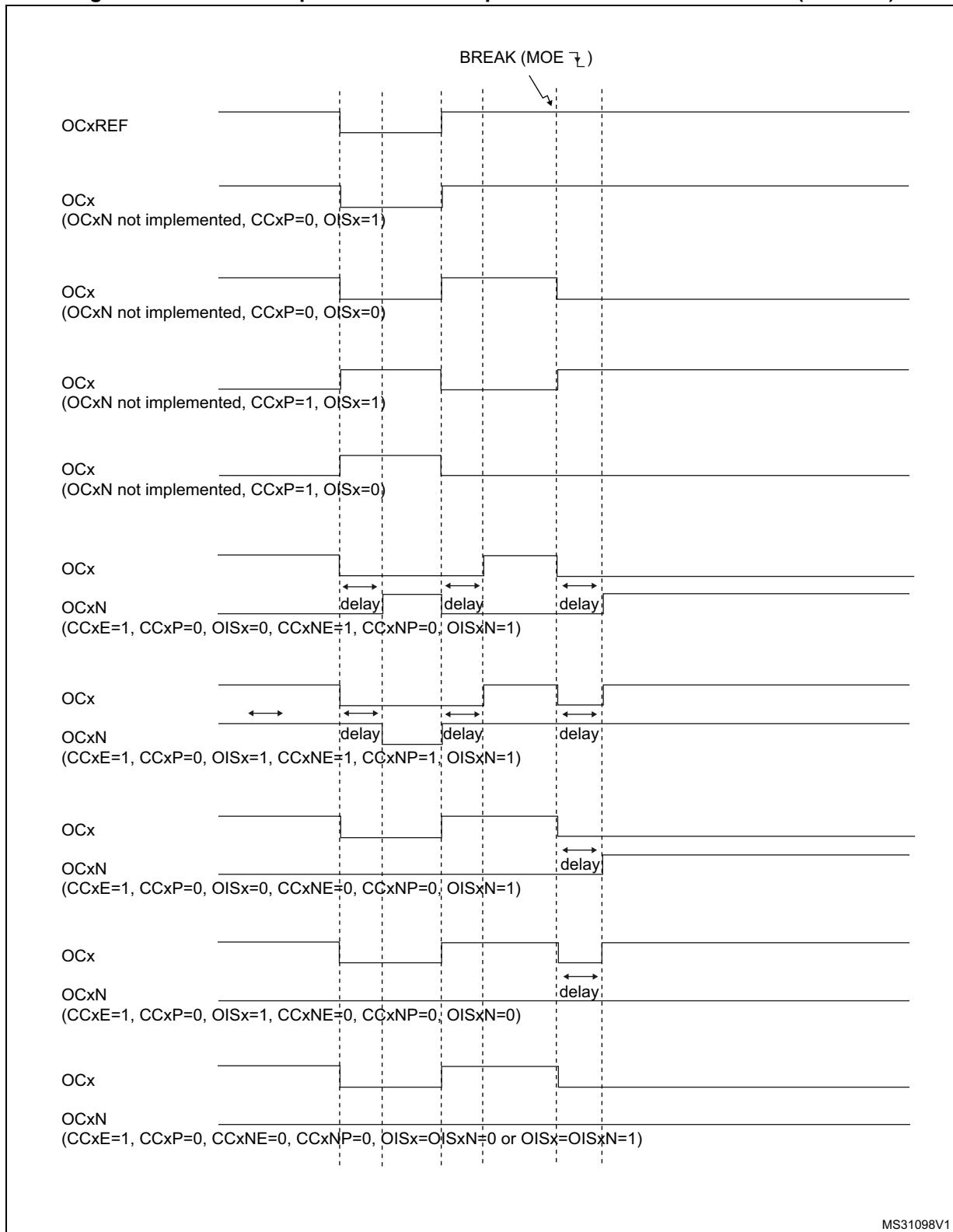
Note:

*The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.*

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx\_BDTR register. Refer to [Section 24.4.20: TIM1 break and dead-time register \(TIM1\\_BDTR\)](#). The LOCK bits can be written only once after an MCU reset.

*Figure 188* shows an example of behavior of the outputs in response to a break.

Figure 188. Various output behavior in response to a break event on BRK (OSSI = 1)



MS31098V1

The two break inputs have different behaviors on timer outputs:

- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in [Table 159](#).

*Note:* BRK2 must only be used with OSSR = OSSI = 1.

**Table 159. Behavior of timer outputs versus BRK/BRK2 inputs**

BRK	BRK2	Timer outputs state	Typical use case	
			OCxN output (low side switches)	OCx output (high side switches)
Active	X	<ul style="list-style-type: none"> <li>– Inactive then forced output state (after a deadtime)</li> <li>– Outputs disabled if OSSI = 0 (control taken over by GPIO logic)</li> </ul>	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

[Figure 189](#) gives an example of OCx and OCxN output behavior in case of active signals on BRK and BRK2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx\_CCER register).

**Figure 189. PWM output state following BRK and BRK2 pins assertion (OSSI=1)**

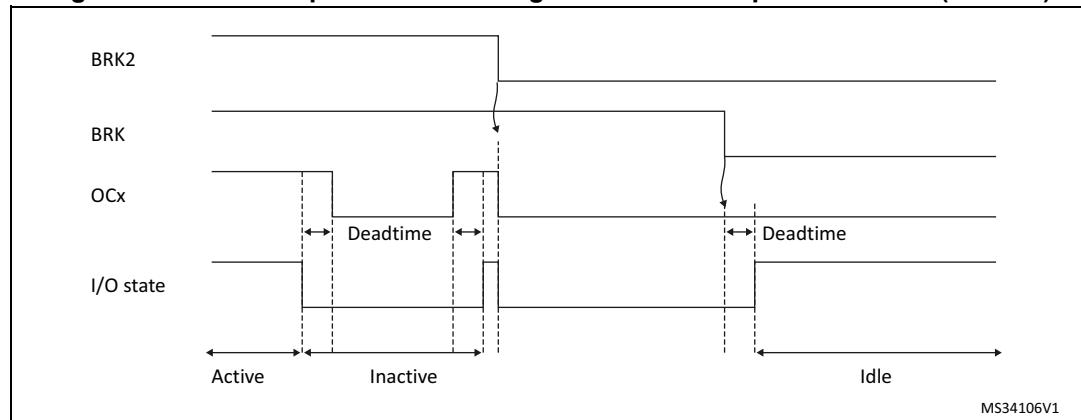
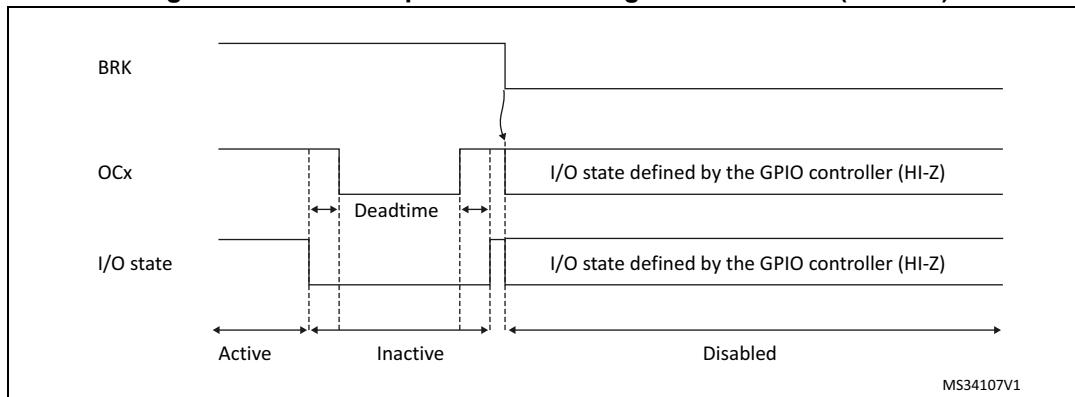


Figure 190. PWM output state following BRK assertion (OSSI=0)



### 24.3.17 Bidirectional break inputs

The TIM1 are featuring bidirectional break I/Os, as represented on [Figure 191](#).

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break and break2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the break and break2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BK(2)E = 1). When a software break event is generated with BK(2)E = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break(2) I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BK(2)DSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see [Table 160](#))

Table 160. Break protection disarming conditions

MOE	BKDIR (BK2DIR)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

### Arming and re-arming break circuitry

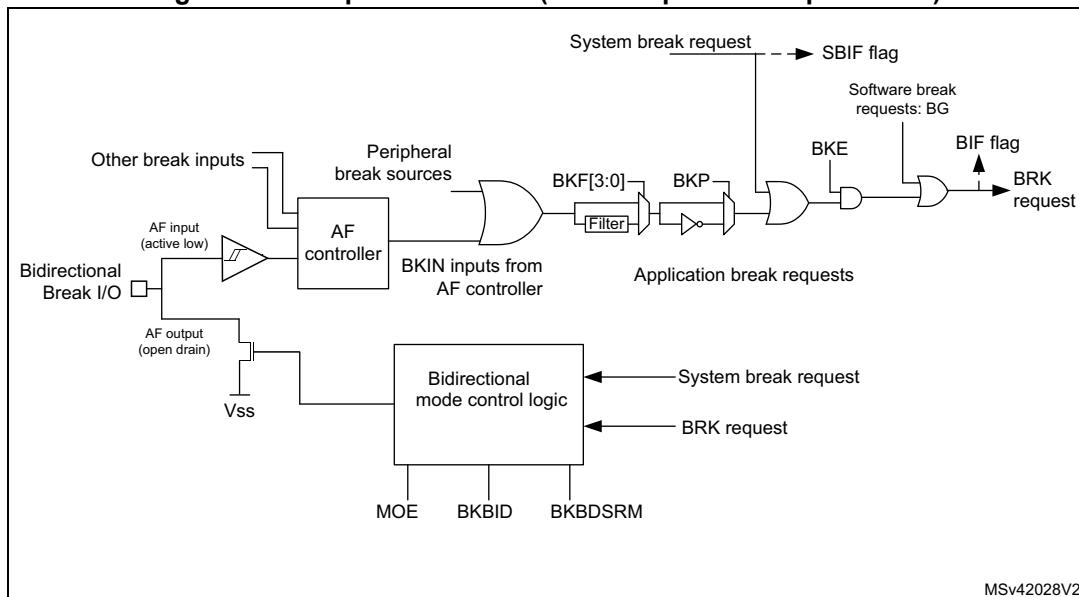
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 191. Output redirection (BRK2 request not represented)



MSv42028V2

### 24.3.18 Clearing the OCxREF signal on an external event

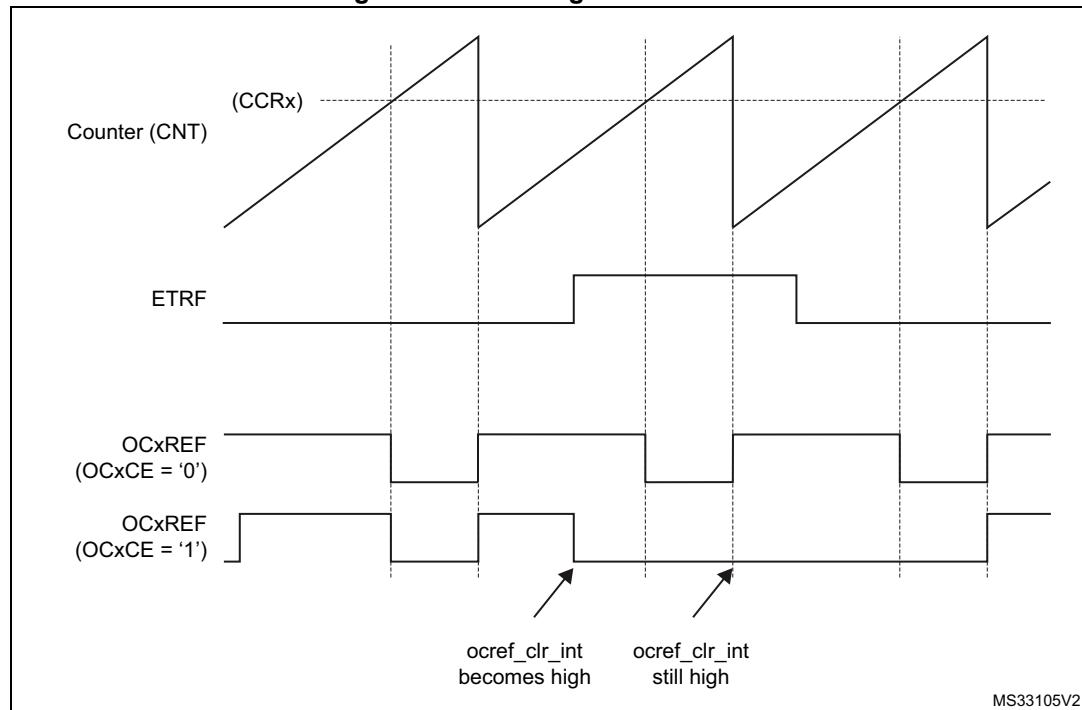
The OCxREF signal of a given channel can be cleared when a high level is applied on the `ocref_clr_int` input (OCxCE enable bit in the corresponding `TIMx_CCMRx` register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode. `ocref_clr_int` input can be selected between the OCREF\_CLR input and ETRF (ETR after the filter) by configuring the OCCS bit in the `TIMx_SMCR` register.

When ETRF is chosen, ETR must be configured as follows:

1. The External Trigger Prescaler should be kept off: bits `ETPS[1:0]` of the `TIMx_SMCR` register set to '00'.
2. The external clock mode 2 must be disabled: bit `ECE` of the `TIMx_SMCR` register set to '0'.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

*Figure 192* shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

**Figure 192. Clearing TIMx OCxREF**



**Note:** *In case of a PWM with a 100% duty cycle (if  $CCRx > ARR$ ), then OCxREF is enabled again at the next counter overflow.*

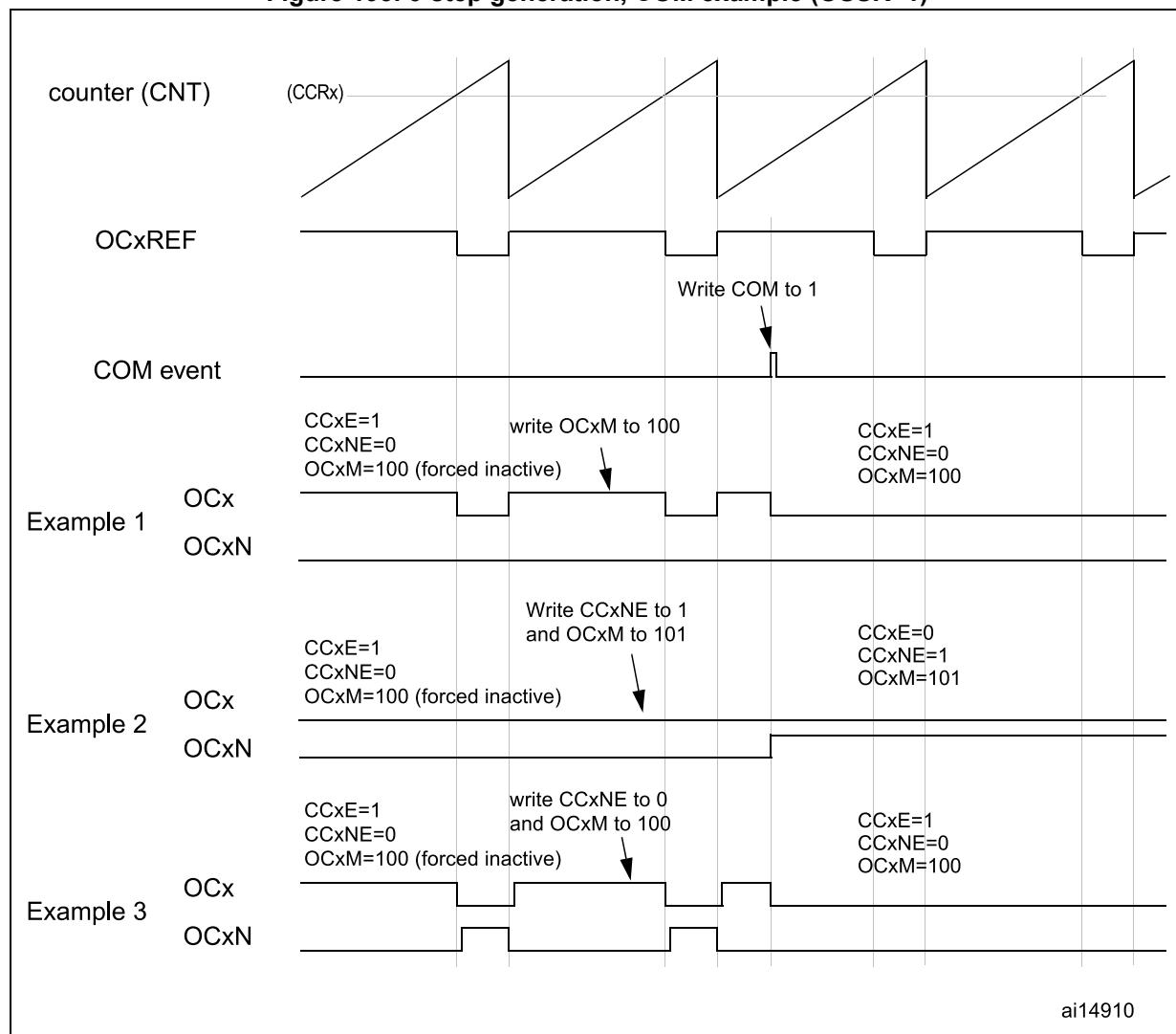
### 24.3.19 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx\_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx\_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx\_DIER register) or a DMA request (if the COMDE bit is set in the TIMx\_DIER register).

The [Figure 193](#) describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

**Figure 193. 6-step generation, COM example (OSSR=1)**



### 24.3.20 One-pulse mode

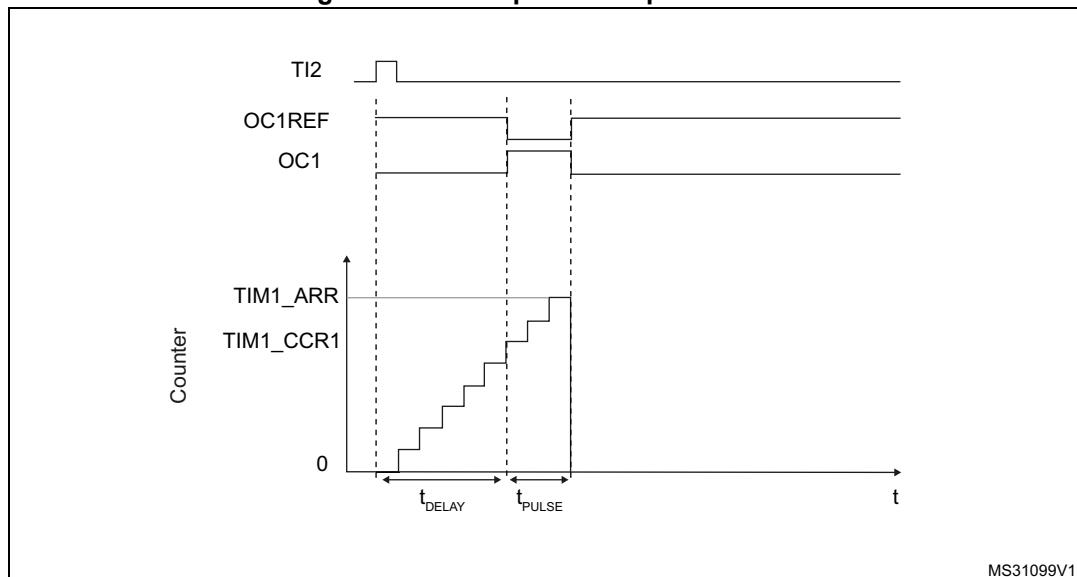
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting:  $CNT < CCRx \leq ARR$  (in particular,  $0 < CCRx$ )
- In downcounting:  $CNT > CCRx$

Figure 194. Example of one pulse mode.



MS31099V1

For example one may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx\_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx\_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx\_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{DELAY}$  is defined by the value written in the  $\text{TIMx\_CCR1}$  register.
- The  $t_{PULSE}$  is defined by the difference between the auto-reload value and the compare value ( $\text{TIMx\_ARR} - \text{TIMx\_CCR1}$ ).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing  $\text{OC1M}=111$  in the  $\text{TIMx\_CCMR1}$  register. Optionally the preload registers can be enabled by writing  $\text{OC1PE}'1'$  in the  $\text{TIMx\_CCMR1}$  register and  $\text{ARPE}$  in the  $\text{TIMx\_CR1}$  register. In this case one has to write the compare value in the  $\text{TIMx\_CCR1}$  register, the auto-reload value in the  $\text{TIMx\_ARR}$  register, generate an update by setting the  $\text{UG}$  bit and wait for external trigger event on  $\text{TI2}$ .  $\text{CC1P}$  is written to '0' in this example.

In our example, the  $\text{DIR}$  and  $\text{CMS}$  bits in the  $\text{TIMx\_CR1}$  register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the  $\text{OPM}$  bit in the  $\text{TIMx\_CR1}$  register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When  $\text{OPM}$  bit in the  $\text{TIMx\_CR1}$  register is set to '0', so the Repetitive Mode is selected.

Particular case:  $\text{OCx}$  fast enable:

In One-pulse mode, the edge detection on  $\text{TIx}$  input set the  $\text{CEN}$  bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{DELAY}$  min we can get.

If one wants to output a waveform with the minimum delay, the  $\text{OCxFE}$  bit can be set in the  $\text{TIMx\_CCMRx}$  register. Then  $\text{OCxRef}$  (and  $\text{OCx}$ ) 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.  $\text{OCxFE}$  acts only if the channel is configured in PWM1 or PWM2 mode.

### 24.3.21 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in [Section 24.3.20](#):

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

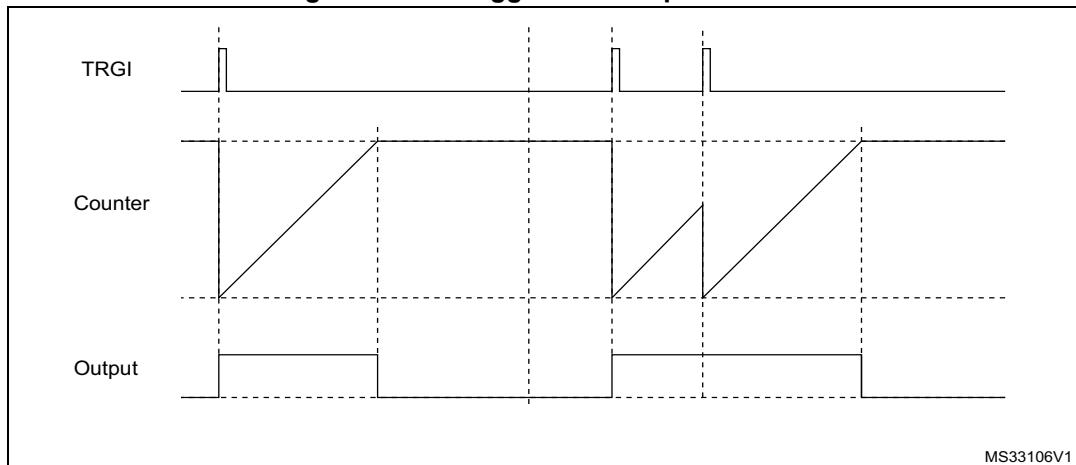
The timer must be in Slave mode, with the bits  $\text{SMS}[3:0] = '1000'$  (Combined Reset + trigger mode) in the  $\text{TIMx\_SMCR}$  register, and the  $\text{OCxM}[3:0]$  bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding  $\text{CCRx}$  must be set to 0 (the  $\text{ARR}$  register sets the pulse length). If the timer is configured in Down-counting mode,  $\text{CCRx}$  must be above or equal to  $\text{ARR}$ .

**Note:** *The  $\text{OCxM}[3:0]$  and  $\text{SMS}[3:0]$  bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.*

*This mode must not be used with center-aligned PWM modes. It is mandatory to have  $\text{CMS}[1:0] = 00$  in  $\text{TIMx\_CR1}$ .*

Figure 195. Retriggerable one pulse mode



#### 24.3.22 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx\_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx\_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to a quadrature encoder. Refer to [Table 161](#). The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx\_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx\_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx\_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx\_ARR must be configured before starting. In the same way, the capture, compare, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

*Note:*

*The prescaler must be set to zero when encoder mode is enabled*

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 161. Counting direction versus encoder signals

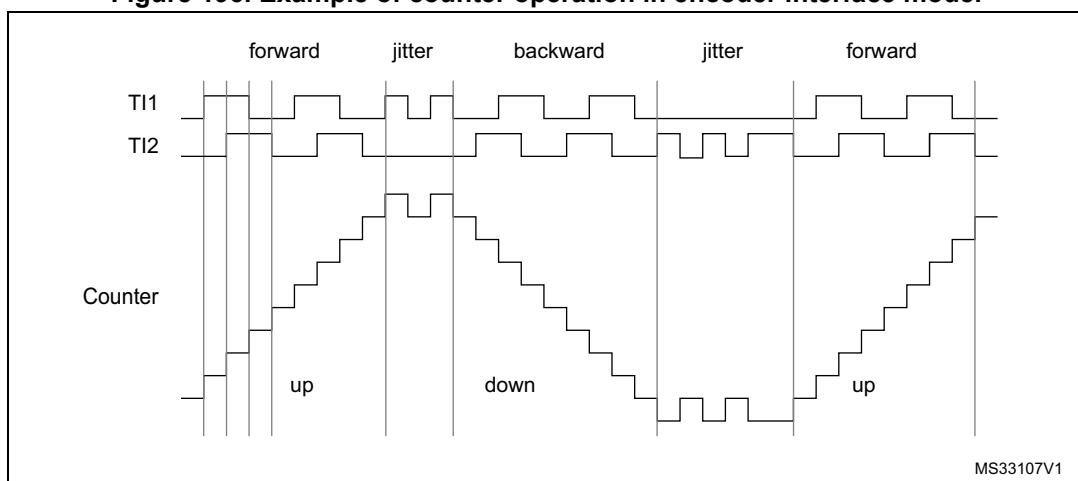
Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The [Figure 196](#) 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:

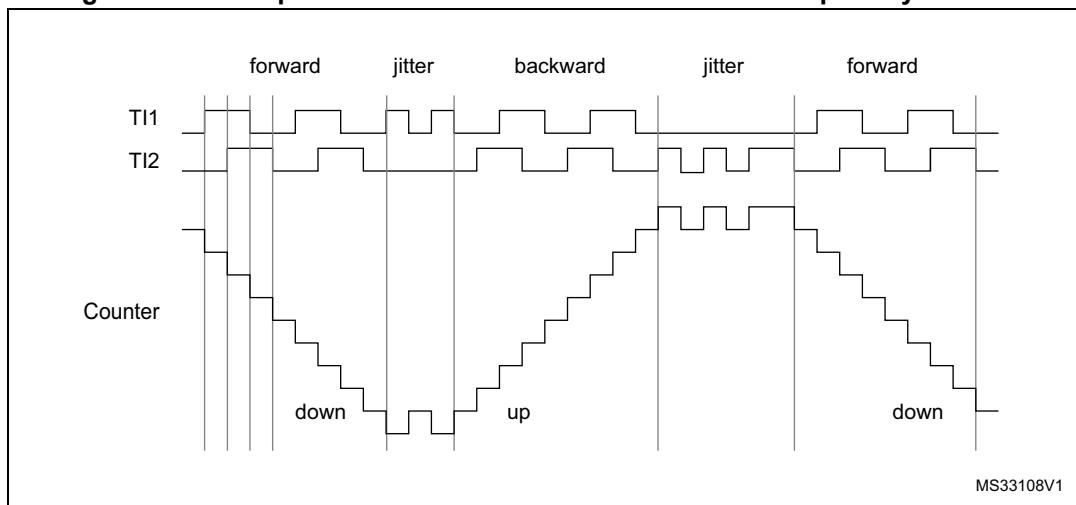
- CC1S='01' (TIMx\_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx\_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P='0' and CC1NP='0' (TIMx\_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0' and CC2NP='0' (TIMx\_CCER register, TI1FP2 non-inverted, TI1FP2=TI2).
- SMS='011' (TIMx\_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx\_CR1 register, Counter enabled).

Figure 196. Example of counter operation in encoder interface mode.



*Figure 197* gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

**Figure 197. Example of encoder interface mode with TI1FP1 polarity inverted.**



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

### 24.3.23 UIF bit remapping

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

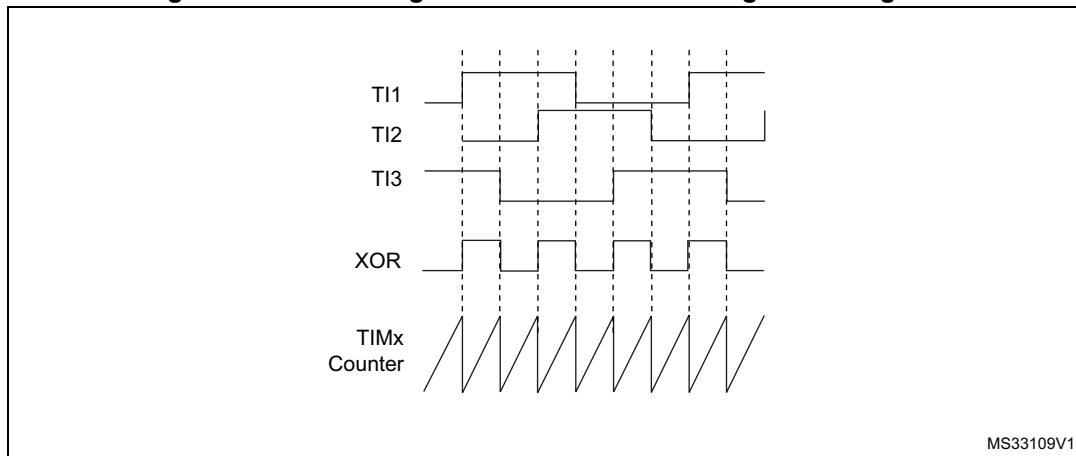
There is no latency between the UIF and UIFCPY flags assertion.

### 24.3.24 Timer input XOR function

The TI1S bit in the TIMx\_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMx\_CH1, TIMx\_CH2 and TIMx\_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per [Figure 198](#) below.

**Figure 198. Measuring time interval between edges on 3 signals**



### 24.3.25 Interfacing with Hall sensors

This is done using the advanced-control timer (TIM1) to generate PWM signals to drive the motor and another timer TIMx (TIM2) referred to as “interfacing timer” in [Figure 199](#). 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 172: Capture/compare channel \(example: channel 1 input stage\) on page 684](#)). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1) through the TRGO output.

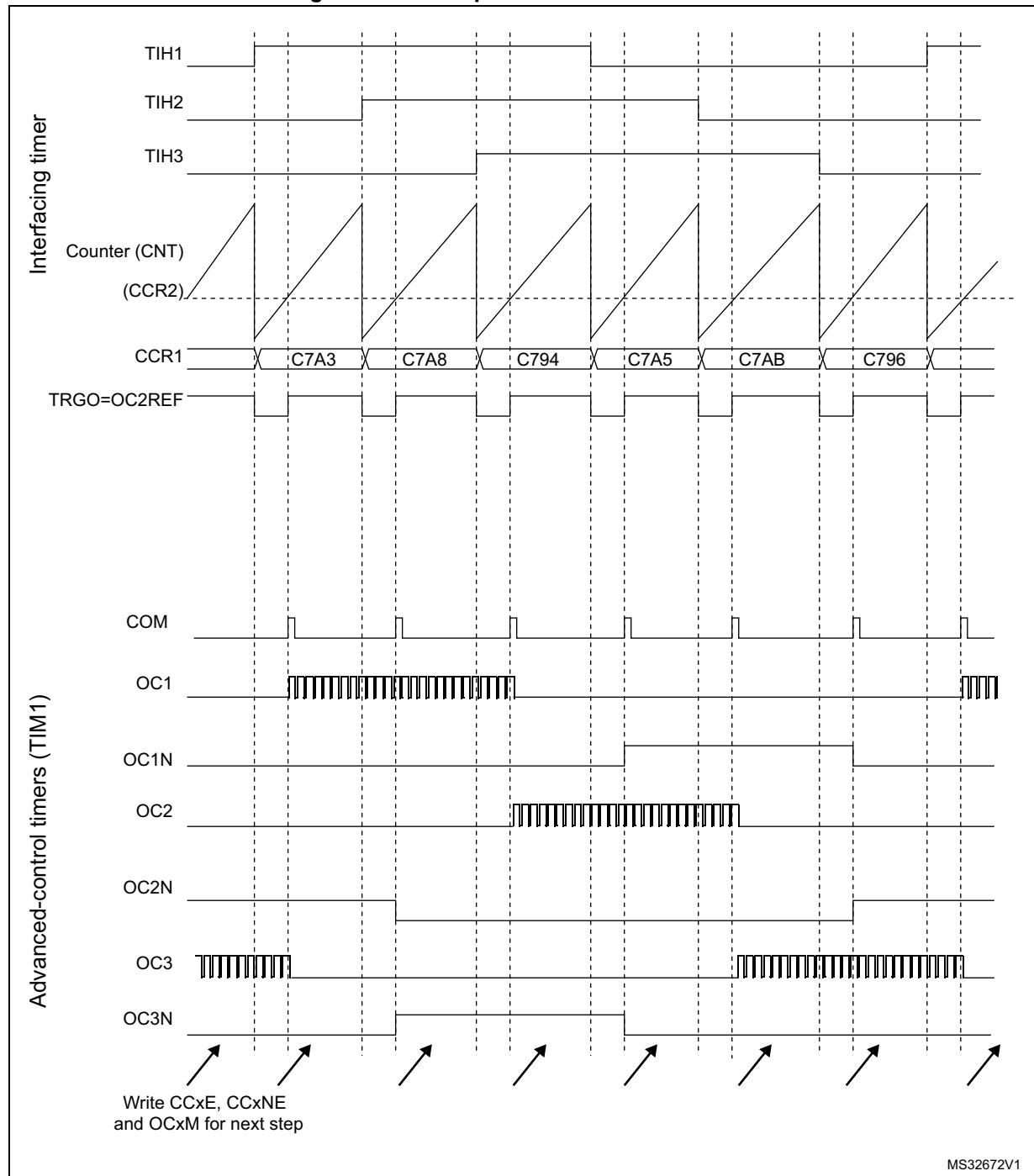
Example: one wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx\_CR2 register to '1',
- Program the time base: write the TIMx\_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx\_CCMR1 register to '01'. The digital filter can also be programmed if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to '111' and the CC2S bits to '00' in the TIMx\_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx\_CR2 register to '101',

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx\_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx\_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The [Figure 199](#) describes this example.

Figure 199. Example of Hall sensor interface



### 24.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to [Section 25.3.19: Timer synchronization](#) for details. They can be synchronized in several modes: Reset mode, Gated mode, and Trigger mode.

#### Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx\_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx\_ARR, TIMx\_CCRx) are updated.

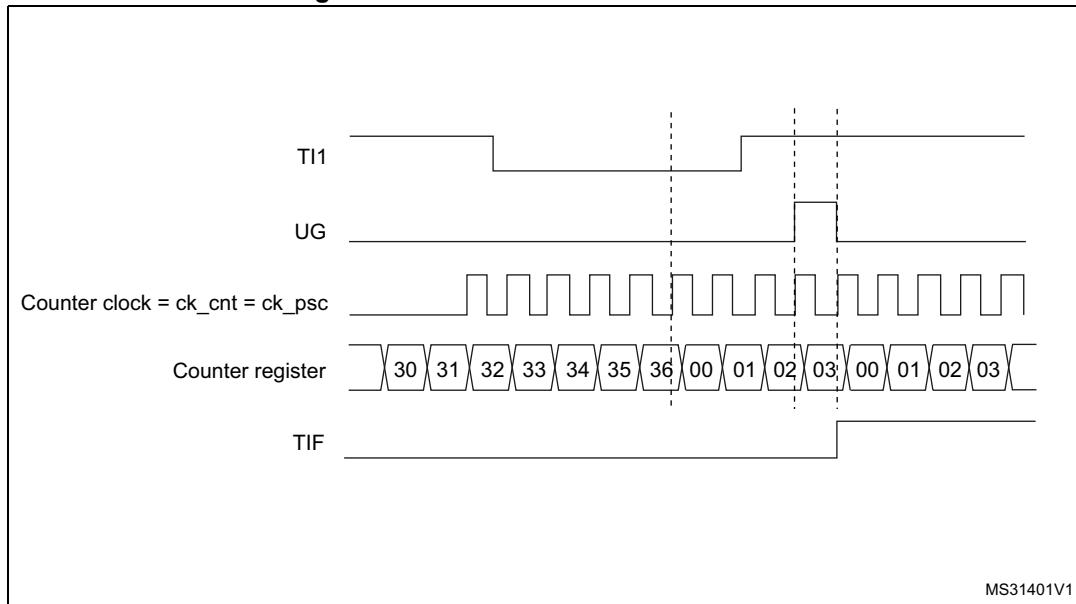
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx\_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx\_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.
- Start the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx\_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx\_DIER register).

The following figure shows this behavior when the auto-reload register TIMx\_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 200. Control circuit in reset mode



MS31401V1

### Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

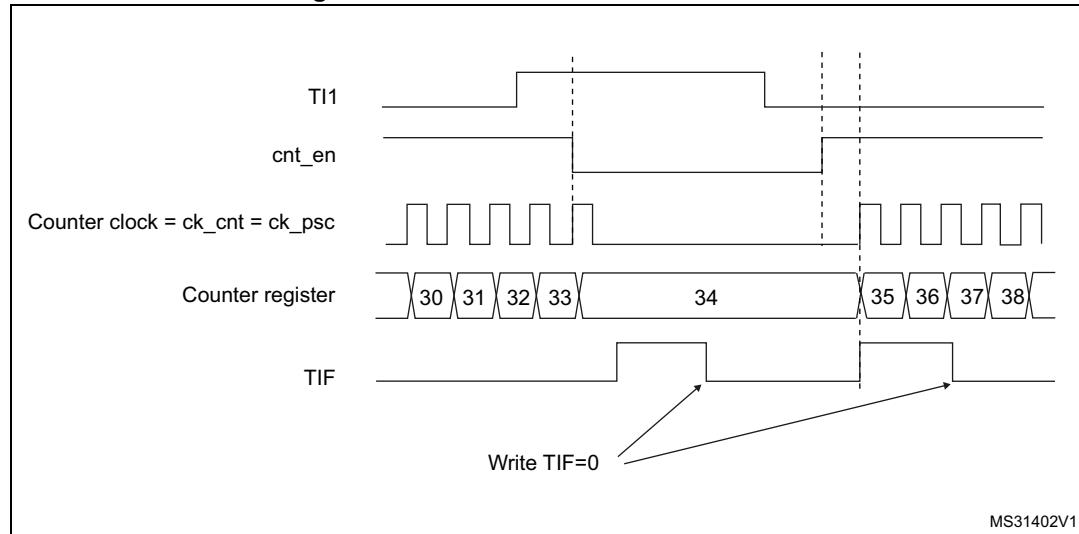
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx\_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx\_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx\_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx\_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

**Figure 201. Control circuit in Gated mode**



### Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx\_CCMR1

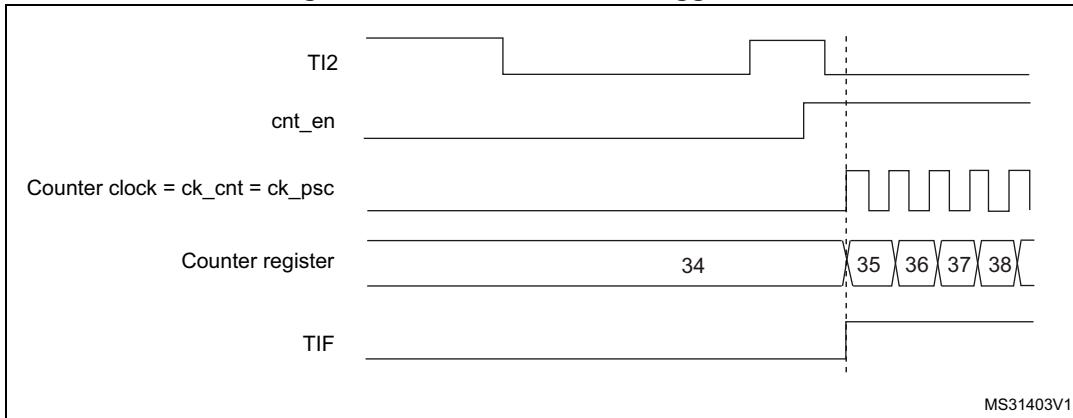
register. Write CC2P=1 and CC2NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).

- Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

**Figure 202. Control circuit in trigger mode**



### Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

### Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx\_SMCR register.

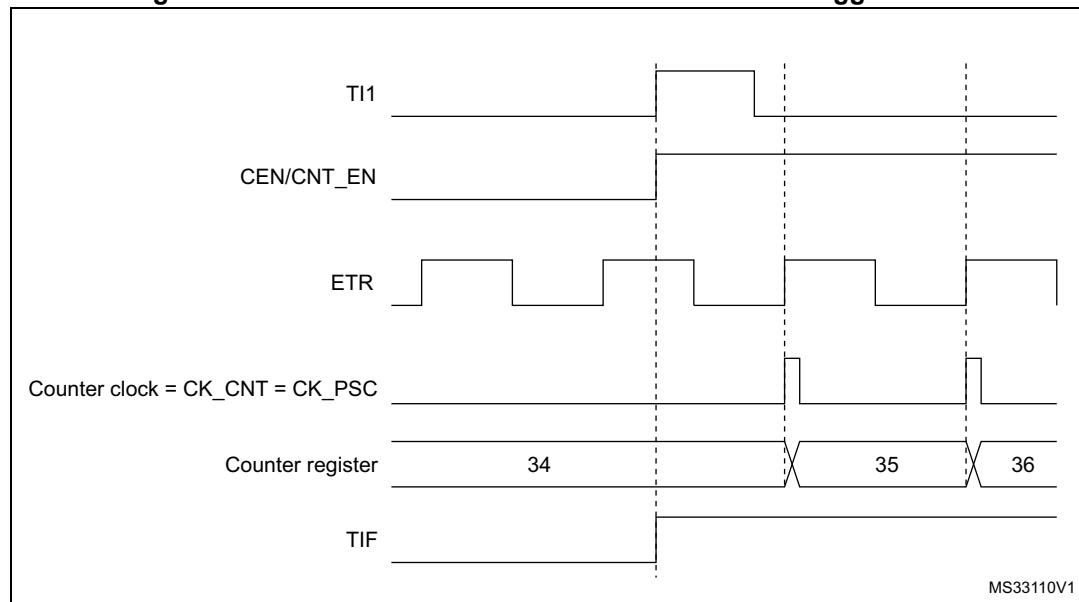
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx\_SMCR register as follows:
  - ETF = 0000: no filter
  - ETPS = 00: prescaler disabled
  - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
  - IC1F = 0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - CC1S = 01 in TIMx\_CCMR1 register to select only the input capture source
  - CC1P = 0 and CC1NP = 0 in TIMx\_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

**Figure 203. Control circuit in external clock mode 2 + trigger mode**



**Note:**

*The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.*

### 24.3.27 ADC synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the TRGO2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx\_CR2 register.

An example of an application for 3-phase motor drives is given in [Figure 183 on page 696](#).

**Note:** *The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.*

**Note:** *The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.*

### 24.3.28 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx\_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx\_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx\_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx\_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register:

Example:

00000: TIMx\_CR1

00001: TIMx\_CR2

00010: TIMx\_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers ( $x = 2, 3, 4$ ) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
  - DMA channel peripheral address is the DMAR register address
  - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
  - Number of data to transfer = 3 (See note below).
  - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:  
DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

*Note:* A null value can be written to the reserved registers.

#### 24.3.29 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on DBG\_TIM1\_STOP configuration bit in DBGMCU module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

## 24.4 TIM1 registers

Refer to for a list of abbreviations used in register descriptions.

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

### 24.4.1 TIM1 control register 1 (TIM1\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK\_INT) frequency and the dead-time and sampling clock ( $t_{DTS}$ ) used by the dead-time generators and the digital filters (ETR, TIx):

00:  $t_{DTS} = t_{CK\_INT}$

01:  $t_{DTS} = 2 * t_{CK\_INT}$

10:  $t_{DTS} = 4 * t_{CK\_INT}$

11: Reserved, do not program this value

*Note:*  $t_{DTS} = 1/f_{DTS}$ ,  $t_{CK\_INT} = 1/f_{CK\_INT}$ .

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx\_ARR register is not buffered

1: TIMx\_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set both when the counter is counting up or down.

*Note:* *Switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1) is not allowed*

Bit 4 **DIR**: Direction

- 0: Counter used as upcounter
- 1: Counter used as downcounter

*Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.*

Bit 3 **OPM**: One pulse mode

- 0: Counter is not stopped at update event
- 1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

- 0: Any of the following events generate an update interrupt or DMA request if enabled.
- These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

- 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

- 0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

- 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

- 0: Counter disabled
- 1: Counter enabled

*Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

## 24.4.2 TIM1 control register 2 (TIM1\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	
								rw	rw	rw	rw		rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]				CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (TRGO2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx\_EGR register is used as trigger output (TRGO2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on TRGO2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT\_EN is used as trigger output (TRGO2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx\_SMCR register).

0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).

0100: **Compare** - OC1REFC signal is used as trigger output (TRGO2)

0101: **Compare** - OC2REFC signal is used as trigger output (TRGO2)

0110: **Compare** - OC3REFC signal is used as trigger output (TRGO2)

0111: **Compare** - OC4REFC signal is used as trigger output (TRGO2)

1000: **Compare** - OC5REFC signal is used as trigger output (TRGO2)

1001: **Compare** - OC6REFC signal is used as trigger output (TRGO2)

1010: **Compare Pulse** - OC4REFC rising or falling edges generate pulses on TRGO2

1011: **Compare Pulse** - OC6REFC rising or falling edges generate pulses on TRGO2

1100: **Compare Pulse** - OC4REFC or OC6REFC rising edges generate pulses on TRGO2

1101: **Compare Pulse** - OC4REFC rising or OC6REFC falling edges generate pulses on TRGO2

1110: **Compare Pulse** - OC5REFC or OC6REFC rising edges generate pulses on TRGO2

1111: **Compare Pulse** - OC5REFC rising or OC6REFC falling edges generate pulses on TRGO2

*Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output Idle state 6 (OC6 output)

Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output Idle state 5 (OC5 output)

Refer to OIS1 bit

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OIS4**: Output Idle state 4 (OC4 output)

Refer to OIS1 bit

Bit 13 **OIS3N**: Output Idle state 3 (OC3N output)

Refer to OIS1N bit

Bit 12 **OIS3**: Output Idle state 3 (OC3 output)

Refer to OIS1 bit

Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)

Refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)

Refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 7 **TI1S**: TI1 selection

0: The TIMx\_CH1 pin is connected to TI1 input

1: The TIMx\_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow selected information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx\_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT\_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx\_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).

100: **Compare** - OC1REFC signal is used as trigger output (TRGO)

101: **Compare** - OC2REFC signal is used as trigger output (TRGO)

110: **Compare** - OC3REFC signal is used as trigger output (TRGO)

111: **Compare** - OC4REFC signal is used as trigger output (TRGO)

*Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI

*Note: This bit acts only on channels that have a complementary output.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).

*Note: This bit acts only on channels that have a complementary output.*

#### 24.4.3 TIM1 slave mode control register (TIM1\_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or ETR is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge.

1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).*

*It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).*

*If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of  $f_{CK\_INT}$  frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

- 00: Prescaler OFF
- 01: ETRP frequency divided by 2
- 10: ETRP frequency divided by 4
- 11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at  $f_{DTS}$
- 0001:  $f_{SAMPLING} = f_{CK\_INT}$ , N=2
- 0010:  $f_{SAMPLING} = f_{CK\_INT}$ , N=4
- 0011:  $f_{SAMPLING} = f_{CK\_INT}$ , N=8
- 0100:  $f_{SAMPLING} = f_{DTS}/2$ , N=6
- 0101:  $f_{SAMPLING} = f_{DTS}/2$ , N=8
- 0110:  $f_{SAMPLING} = f_{DTS}/4$ , N=6
- 0111:  $f_{SAMPLING} = f_{DTS}/4$ , N=8
- 1000:  $f_{SAMPLING} = f_{DTS}/8$ , N=6
- 1001:  $f_{SAMPLING} = f_{DTS}/8$ , N=8
- 1010:  $f_{SAMPLING} = f_{DTS}/16$ , N=5
- 1011:  $f_{SAMPLING} = f_{DTS}/16$ , N=6
- 1100:  $f_{SAMPLING} = f_{DTS}/16$ , N=8
- 1101:  $f_{SAMPLING} = f_{DTS}/32$ , N=5
- 1110:  $f_{SAMPLING} = f_{DTS}/32$ , N=6
- 1111:  $f_{SAMPLING} = f_{DTS}/32$ , N=8

Bit 7 **MSM**: Master/slave mode

- 0: No action
- 1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

- 00000: Internal Trigger 0 (ITR0)
- 00001: Internal Trigger 1 (ITR1)
- 00010: Internal Trigger 2 (ITR2)
- 00011: Internal Trigger 3 (ITR3)
- 00100: TI1 Edge Detector (TI1F\_ED)
- 00101: Filtered Timer Input 1 (TI1FP1)
- 00110: Filtered Timer Input 2 (TI2FP2)
- 00111: External Trigger input (ETRF)
- Others: Reserved

See [Table 162: TIM1 internal trigger connection on page 728](#) for more details on ITRx meaning for each Timer.

*Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.*

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source.

- 0: OCREF\_CLR\_INT is connected to the OCREF\_CLR input
- 1: OCREF\_CLR\_INT is connected to ETRF

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

0000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Codes above 1000: Reserved.

*Note: The gated mode must not be used if TI1F\_ED is selected as the trigger input (TS=00100). Indeed, TI1F\_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.*

*Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.*

**Table 162. TIM1 internal trigger connection**

Slave TIM	ITR0 (TS = 00000)	ITR1 (TS = 00001)	ITR2 (TS = 00010)	ITR3 (TS = 00011)
TIM1	-	TIM2	-	TIM17 OC1

#### 24.4.4 TIM1 DMA/interrupt enable register (TIM1\_DIER)

Address offset: 0x0C

Reset value: 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	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

#### 24.4.5 TIM1 status register (TIM1\_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF
														rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	SBIFF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0													

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag

Refer to CC1IF description (Note: Channel 6 can only be configured as output)

Bit 16 **CC5IF**: Compare 5 interrupt flag

Refer to CC1IF description (Note: Channel 5 can only be configured as output)

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIFF**: System Break interrupt flag

This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.

This flag must be reset to re-start PWM operation.

0: No break event occurred.

1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx\_DIER register.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag

Refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag

Refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag

Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx\_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx\_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx\_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

**If channel CC1 is configured as output:** this flag is set when the content of the counter TIMx\_CNT matches the content of the TIMx\_CCR1 register. When the content of TIMx\_CCR1 is greater than the content of TIMx\_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx\_CR1 register for the full description.

**If channel CC1 is configured as input:** this bit is set when counter value has been captured in TIMx\_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx\_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx\_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx\_EGR register, if URS=0 and UDIS=0 in the TIMx\_CR1 register.
- When CNT is reinitialized by a trigger event (refer to [Section 24.4.3: TIM1 slave mode control register \(TIM1\\_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx\_CR1 register.

## 24.4.6 TIM1 event generation register (TIM1\_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG						

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx\_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: When CCPC bit is set, it allows CCxE, CCxNE and OCxM bits to be updated.

*Note: This bit acts only on channels having a complementary output.*

Bit 4 **CC4G**: Capture/Compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

**If channel CC1 is configured as output:**

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

**If channel CC1 is configured as input:**

The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. The prescaler internal counter is also cleared (the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx\_ARR) if DIR=1 (downcounting).

#### 24.4.7 TIM1 capture/compare mode register 1 [alternate] (TIM1\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

##### Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]					IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Refer to IC1F[3:0] description.

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Refer to IC1PSC[1:0] description.

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx\_CCER).*

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at  $f_{DTS}$

0001:  $f_{SAMPLING} = f_{CK\_INT}$ , N=2

0010:  $f_{SAMPLING} = f_{CK\_INT}$ , N=4

0011:  $f_{SAMPLING} = f_{CK\_INT}$ , N=8

0100:  $f_{SAMPLING} = f_{DTS}/2$ , N=6

0101:  $f_{SAMPLING} = f_{DTS}/2$ , N=8

0110:  $f_{SAMPLING} = f_{DTS}/4$ , N=6

0111:  $f_{SAMPLING} = f_{DTS}/4$ , N=8

1000:  $f_{SAMPLING} = f_{DTS}/8$ , N=6

1001:  $f_{SAMPLING} = f_{DTS}/8$ , N=8

1010:  $f_{SAMPLING} = f_{DTS}/16$ , N=5

1011:  $f_{SAMPLING} = f_{DTS}/16$ , N=6

1100:  $f_{SAMPLING} = f_{DTS}/16$ , N=8

1101:  $f_{SAMPLING} = f_{DTS}/32$ , N=5

1110:  $f_{SAMPLING} = f_{DTS}/32$ , N=6

1111:  $f_{SAMPLING} = f_{DTS}/32$ , N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx\_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).*

#### 24.4.8 TIM1 capture/compare mode register 1 [alternate] (TIM1\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

#### Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output Compare 2 clear enable

Refer to OC1CE description.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode

Refer to OC1M[3:0] description.

Bit 11 **OC2PE**: Output Compare 2 preload enable

Refer to OC1PE description.

Bit 10 **OC2FE**: Output Compare 2 fast enable

Refer to OC1FE description.

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx\_CCER).*

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: OC1Ref is not affected by the ocref\_clr\_int signal

1: OC1Ref is cleared as soon as a High level is detected on ocref\_clr\_int signal  
(OCREF\_CLR input or ETRF input)

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx\_CNT<TIMx\_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx\_CNT>TIMx\_CCR1 else active (OC1REF='1').

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx\_CNT<TIMx\_CCR1 else active. In downcounting, channel 1 is active as long as TIMx\_CNT>TIMx\_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

*Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

*Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

*Note: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.*

*Note: The OC1M[3] bit is not contiguous, located in bit 16.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx\_CCR1 enabled. Read/Write operations access the preload register. TIMx\_CCR1 preload value is loaded in the active register at each update event.

*Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx\_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: 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).*

#### 24.4.9 TIM1 capture/compare mode register 2 [alternate] (TIM1\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

**Input capture mode:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Refer to IC1F[3:0] description.

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Refer to IC1PSC[1:0] description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx\_CCER).*

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Refer to IC1F[3:0] description.

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Refer to IC1PSC[1:0] description.

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx\_CCER).*

#### 24.4.10 TIM1 capture/compare mode register 2 [alternate] (TIM1\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

##### Output compare mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable  
Refer to OC1CE description.

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode  
Refer to OC3M[3:0] description.

Bit 11 **OC4PE**: Output compare 4 preload enable  
Refer to OC1PE description.

Bit 10 **OC4FE**: Output compare 4 fast enable  
Refer to OC1FE description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx\_CCER).*

Bit 7 **OC3CE**: Output compare 3 clear enable  
Refer to OC1CE description.

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode  
Refer to OC1M[3:0] description.

Bit 3 **OC3PE**: Output compare 3 preload enable  
Refer to OC1PE description.

Bit 2 **OC3FE**: Output compare 3 fast enable  
Refer to OC1FE description.

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx\_CCER).*

#### 24.4.11 TIM1 capture/compare enable register (TIM1\_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/Compare 6 output polarity

Refer to CC1P description

Bit 20 **CC6E**: Capture/Compare 6 output enable

Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/Compare 5 output polarity

Refer to CC1P description

Bit 16 **CC5E**: Capture/Compare 5 output enable

Refer to CC1E description

Bit 15 **CC4NP**: Capture/Compare 4 complementary output polarity

Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output polarity

Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable

Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity

Refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable

Refer to CC1NE description

Bit 9 **CC3P**: Capture/Compare 3 output polarity

Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable

Refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity

Refer to CC1NP description

Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable

Refer to CC1NE description

Bit 5 **CC2P**: Capture/Compare 2 output polarity  
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable  
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

**CC1 channel configured as output:**

- 0: OC1N active high.
- 1: OC1N active low.

**CC1 channel configured as input:**

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S="00" (channel configured as output).*

*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

- 0: Off - OC1N is not active. OC1N level is then function of MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.
- 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.

*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 1 **CC1P**: Capture/Compare 1 output polarity

- 0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
  - 1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
- When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: The configuration is reserved, it must not be used.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**When CC1 channel is configured as output**, the OC1 level depends on MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to [Table 163](#) for details.

*Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

**Table 163. Output control bits for complementary OCx and OCxN channels with break feature**

Control bits					Output states <sup>(1)</sup>	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) OCx=0, OCxN=0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN = OCxREF xor CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN = OCxREF x or CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Off-State (output enabled with inactive state) OCxN=CCxNP
0	X	0	X	X	Output disabled (not driven by the timer: Hi-Z).	
		0	0	0		
		0	1	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered).  Then (this is valid only if BRK is triggered), if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCx and OCxN both in active state (may cause a short circuit when driving switches in half-bridge configuration). <b>Note:</b> BRK2 can only be used if OSSI = OSSR = 1.	
		1	0	0		
		1	1	1		
		1	1	1		

- When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

**Note:** The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

#### 24.4.12 TIM1 counter (TIM1\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.														
r															
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx\_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

#### 24.4.13 TIM1 prescaler (TIM1\_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK\_CNT) is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

#### 24.4.14 TIM1 auto-reload register (TIM1\_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 24.3.1: Time-base unit on page 664](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

#### 24.4.15 TIM1 repetition counter register (TIM1\_RCR)

Address offset: 0x30

Reset value: 0x0000

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

Bits 15:0 **REP[15:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP\_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP\_CNT is reloaded with REP value only at the repetition update event U\_RC, any write to the TIMx\_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:  
the number of PWM periods in edge-aligned mode  
the number of half PWM period in center-aligned mode.

#### 24.4.16 TIM1 capture/compare register 1 (TIM1\_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

**If channel CC1 is configured as output:** CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC1 output.

**If channel CC1 is configured as input:** CR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx\_CCR1 register is read-only and cannot be programmed.

#### 24.4.17 TIM1 capture/compare register 2 (TIM1\_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

**If channel CC2 is configured as output:** CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC2 output.

**If channel CC2 is configured as input:** CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx\_CCR2 register is read-only and cannot be programmed.

#### 24.4.18 TIM1 capture/compare register 3 (TIM1\_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

**If channel CC3 is configured as output:** CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC3 output.

**If channel CC3 is configured as input:** CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx\_CCR3 register is read-only and cannot be programmed.

#### 24.4.19 TIM1 capture/compare register 4 (TIM1\_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

**If channel CC4 is configured as output:** CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC4 output.

**If channel CC4 is configured as input:** CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx\_CCR4 register is read-only and cannot be programmed.

#### 24.4.20 TIM1 break and dead-time register (TIM1\_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2DSRM	BK2DSRM	BK2P	BK2E	BK2F[3:0]							
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

**Note:** As the bits BK2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSR, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx\_BDTR register.

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

Bit 29 **BK2BID**: Break2 bidirectional

Refer to BKBID description

Bit 28 **BKBID**: Break Bidirectional

- 0: Break input BRK in input mode
- 1: Break input BRK in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 27 **BK2DSRM**: Break2 Disarm

Refer to BKDSRM description

Bit 26 **BKDSRM**: Break Disarm

- 0: Break input BRK is armed
- 1: Break input BRK is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 25 **BK2P**: Break 2 polarity

- 0: Break input BRK2 is active low
- 1: Break input BRK2 is active high

*Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 24 **BK2E**: Break 2 enable

*Note: The must only be used with OSSR = OSSI = 1.*

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bit-field defines the frequency used to sample BRK2 input and the length of the digital filter applied to BRK2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, BRK2 acts asynchronously
- 0001:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , N=2
- 0010:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , N=4
- 0011:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , N=8
- 0100:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$ , N=6
- 0101:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$ , N=8
- 0110:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$ , N=6
- 0111:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$ , N=8
- 1000:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$ , N=6
- 1001:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$ , N=8
- 1010:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$ , N=5
- 1011:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$ , N=6
- 1100:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$ , N=8
- 1101:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$ , N=5
- 1110:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$ , N=6
- 1111:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$ , N=8

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, BRK acts asynchronously
- 0001:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , N=2
- 0010:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , N=4
- 0011:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , N=8
- 0100:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$ , N=6
- 0101:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$ , N=8
- 0110:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$ , N=6
- 0111:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$ , N=8
- 1000:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$ , N=6
- 1001:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$ , N=8
- 1010:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$ , N=5
- 1011:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$ , N=6
- 1100:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$ , N=8
- 1101:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$ , N=5
- 1110:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$ , N=6
- 1111:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$ , N=8

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (BRK or BRK2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled

In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSS1 bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx\_CCER register).

See OC/OCN enable description for more details ([Section 24.4.11: TIM1 capture/compare enable register \(TIM1\\_CCER\)](#)).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs BRK and BRK2 is active)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk\_acth and BKIN sources, as per [Figure 187: Break and Break2 circuitry overview](#)).

0: Break function disabled

1: Break function enabled

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details ([Section 24.4.11: TIM1 capture/compare enable register \(TIM1\\_CCER\)](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs.

See OC/OCN enable description for more details ([Section 24.4.11: TIM1 capture/compare enable register \(TIM1\\_CCER\)](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx\_BDTR register, OISx and OISxN bits in TIMx\_CR2 register and BK2BID, BKBD, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSS1, OSSR and DTG[7:0] bits in TIMx\_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx\_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSS1 bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx\_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

*Note: The LOCK bits can be written only once after the reset. Once the TIMx\_BDTR register has been written, their content is frozen until the next reset.*

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx => DT = DTG[7:0] x t<sub>DTG</sub> with t<sub>DTG</sub> = t<sub>DTS</sub>.

DTG[7:5] = 10x => DT = (64 + DTG[5:0]) x t<sub>DTG</sub> with t<sub>DTG</sub> = 2 x t<sub>DTS</sub>.

DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t<sub>DTG</sub> with t<sub>DTG</sub> = 8 x t<sub>DTS</sub>.

DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t<sub>DTG</sub> with t<sub>DTG</sub> = 16 x t<sub>DTS</sub>.

Example if t<sub>DTS</sub> = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

*Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

#### 24.4.21 TIM1 DMA control register (TIM1\_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]				Res.	Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer  
00001: 2 transfers  
00010: 3 transfers

...  
10001: 18 transfers

**Example:** Let us consider the following transfer: DBL = 7 bytes & DBA = TIMx\_CR1.

– If DBL = 7 bytes and DBA = TIMx\_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

(TIMx\_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx\_CR1 address) + DBA, which gives us the address from/to which the data is copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx\_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

- If the DMA Data Size is configured in half-words, 16-bit data is transferred to each of the 7 registers.
- If the DMA Data Size is configured in bytes, the data is also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

**Example:**

00000: TIMx\_CR1,  
00001: TIMx\_CR2,  
00010: TIMx\_SMCR,

...

#### 24.4.22 TIM1 DMA address for full transfer (TIM1\_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx\_CR1 address) + (DBA + DMA index) x 4

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

#### 24.4.23 TIM1 option register 1 (TIM1\_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TI1_RMP	Res.	Res.	TIM1_ETR_ADC_RMP[1:0]											
											rw			rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TI1\_RMP**: Input Capture 1 remap

0: TIM1 input capture 1 is connected to I/O

1: TIM1 input capture 1 is connected to COMP1 output

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

Bits 1:0 **TIM1\_ETR\_ADC\_RMP[1:0]**: TIM1\_ETR\_ADC remapping capability

00: TIM1\_ETR is not connected to ADC AWDx (must be selected when the ETR comes from the ETR input pin)

01: TIM1\_ETR is connected to ADC AWD1

10: TIM1\_ETR is connected to ADC AWD2

11: TIM1\_ETR is connected to ADC AWD3

*Note: ADC AWDx sources are 'ORed' with the TIM1\_ETR input signals. When ADC AWDx is used, it is necessary to make sure that the corresponding TIM1\_ETR input pin is not enabled in the alternate function controller.*

#### 24.4.24 TIM1 capture/compare mode register 3 (TIM1\_CCMR3)

Address offset: 0x54

Reset value: 0x0000 0000

The channels 5 and 6 can only be configured in output.

**Output compare mode:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6CE	OC6M[2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

Refer to OC1CE description.

Bits 24, 14, 13, 12 **OC6M[3:0]**: Output compare 6 mode

Refer to OC1M description.

Bit 11 **OC6PE**: Output compare 6 preload enable

Refer to OC1PE description.

Bit 10 **OC6FE**: Output compare 6 fast enable

Refer to OC1FE description.

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

Bit 7 **OC5CE**: Output compare 5 clear enable

Refer to OC1CE description.

Bits 16, 6, 5, 4 **OC5M[3:0]**: Output compare 5 mode

Refer to OC1M description.

Bit 3 **OC5PE**: Output compare 5 preload enable

Refer to OC1PE description.

Bit 2 **OC5FE**: Output compare 5 fast enable

Refer to OC1FE description.

Bits 1:0 Reserved, must be kept at reset value.

#### 24.4.25 TIM1 capture/compare register 5 (TIM1\_CCR5)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.												
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group Channel 5 and Channel 3

Distortion on Channel 3 output:

0: No effect of OC5REF on OC3REFC

1: OC3REFC is the logical AND of OC3REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).

*Note: it is also possible to apply this distortion on combined PWM signals.*

Bit 30 **GC5C2**: Group Channel 5 and Channel 2

Distortion on Channel 2 output:

0: No effect of OC5REF on OC2REFC

1: OC2REFC is the logical AND of OC2REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

*Note: it is also possible to apply this distortion on combined PWM signals.*

Bit 29 **GC5C1**: Group Channel 5 and Channel 1

Distortion on Channel 1 output:

0: No effect of OC5REF on OC1REFC5

1: OC1REFC is the logical AND of OC1REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

*Note: it is also possible to apply this distortion on combined PWM signals.*

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR5[15:0]**: Capture/Compare 5 value

CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC5 output.

#### 24.4.26 TIM1 capture/compare register 6

##### (TIM1\_CCR6)

Address offset: 0x5C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR6[15:0]**: Capture/Compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC6 output.

### 24.4.27 TIM1 alternate function option register 1 (TIM1\_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	BK CMP2P	BK CMP1P	BKINP	Res.	Res.	Res.	Res.	Res.	Res.	BK CMP2E	BK CMP1E	BKINE
rw	rw			rw	rw	rw							rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.

0000: ETR legacy mode

0001: COMP1 output

0010: COMP2 output

Others: Reserved

*Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

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

Bit 11 **BKCM2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: COMP2 input polarity is inverted (active high if BKP=0, active low if BKP=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **BKCM1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: COMP1 input polarity is inverted (active high if BKP=0, active low if BKP=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: BKIN input polarity is inverted (active high if BKP=0, active low if BKP=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCM2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

- 0: COMP2 input disabled
- 1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 1 **BKCM1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

- 0: COMP1 input disabled
- 1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

- 0: BKIN input disabled
- 1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Note:** Refer to [Figure 166: TIM1 ETR input circuitry](#) and to [Figure 187: Break and Break2 circuitry overview](#).

#### 24.4.28 TIM1 Alternate function register 2 (TIM1\_AF2)

Address offset: 0x64

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BK2 CMP2 P	BK2 CMP1 P	BK2 INP	Res.	Res.	Res.	Res.	Res.	Res.	BK2 CMP2E	BK2 CMP1E	BK2INE
				rw	rw	rw							rw	rw	rw

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

Bit 11 **BK2CMP2P**: BRK2 COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BK2P polarity bit.

- 0: COMP2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)
- 1: COMP2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **BK2CMP1P**: BRK2 COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BK2P polarity bit.

- 0: COMP1 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)
- 1: COMP1 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 9 **BK2INP**: BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

- 0: BKIN2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)
- 1: BKIN2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BK2CMP2E**: BRK2 COMP2 enable

This bit enables the COMP2 for the timer's BRK2 input. COMP2 output is 'ORed' with the other BRK2 sources.

- 0: COMP2 input disabled
- 1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 1 **BK2CMP1E**: BRK2 COMP1 enable

This bit enables the COMP1 for the timer's BRK2 input. COMP1 output is 'ORed' with the other BRK2 sources.

- 0: COMP1 input disabled
- 1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BK2INE**: BRK2 BKIN input enable

This bit enables the BKIN2 alternate function input for the timer's BRK2 input. BKIN2 input is 'ORed' with the other BRK2 sources.

- 0: BKIN2 input disabled
- 1: BKIN2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Note:** Refer to [Figure 187: Break and Break2 circuitry overview](#).

### 24.4.29 TIM1 timer input selection register (TIM1\_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

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

Bits 27:24 **TI4SEL[3:0]**: selects TI4[0] to TI4[15] input

0000: TIM1\_CH4 input

Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: selects TI3[0] to TI3[15] input

0000: TIM1\_CH3 input

Others: Reserved

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

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input

0000: TIM1\_CH2 input

Others: Reserved

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

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM1\_CH1 input

Others: Reserved

### 24.4.30 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 164. TIM1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
0x00	TIM1_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
		Reset value																																				
0x04	TIM1_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
		Reset value																																				
0x08	TIM1_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
		Reset value																																				
0x0C	TIM1_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
		Reset value																																				
0x10	TIM1_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
		Reset value																																				
0x14	TIM1_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
		Reset value																																				
0x18	TIM1_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
		Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIM1_CCMR2 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		Reset value	0	0	0	0	0	0	0	0	0	0	0	0	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	TIM1_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 164. TIM1 register map and reset values (continued)

Offset	Register name	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x24	<b>TIM1_CNT</b>	0	UIFCP	31	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													
	Reset value																																			
0x28	<b>TIM1_PSC</b>																																			
	Reset value																																			
0x2C	<b>TIM1_ARR</b>																																			
	Reset value																																			
0x30	<b>TIM1_RCR</b>																																			
	Reset value																																			
0x34	<b>TIM1_CCR1</b>																																			
	Reset value																																			
0x38	<b>TIM1_CCR2</b>																																			
	Reset value																																			
0x3C	<b>TIM1_CCR3</b>																																			
	Reset value																																			
0x40	<b>TIM1_CCR4</b>																																			
	Reset value																																			
0x44	<b>TIM1_BDTR</b>																																			
	Reset value																																			
0x48	<b>TIM1_DCR</b>																																			
	Reset value																																			
0x4C	<b>TIM1_DMAR</b>																																			
	Reset value																																			
0x50	<b>TIM1_OR1</b>																																			
	Reset value																																			

Table 164. TIM1 register map and reset values (continued)

Offset	Register name	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
0x54	TIM1_CCMR3 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0	GC5C3	0	GC5C2	0	GC5C1	0	Res.	0	GC5C3	0	GC5C2	0	GC5C1	0	Res.	0	GC5C3	0	GC5C2	0	GC5C1	0	Res.	0	GC5C3	0	GC5C2	0	GC5C1	0	Res.	0	GC5C3	0	GC5C2	0	GC5C1	0	Res.
0x58	TIM1_CCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	TIM1_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	TIM1_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	TIM1_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	TIM1_TISEL	TI4SEL[3:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 25 General-purpose timer (TIM2)

### 25.1 TIM2 introduction

The general-purpose timer TIM2 consists of a 32-bit auto-reload counter driven by a programmable prescaler.

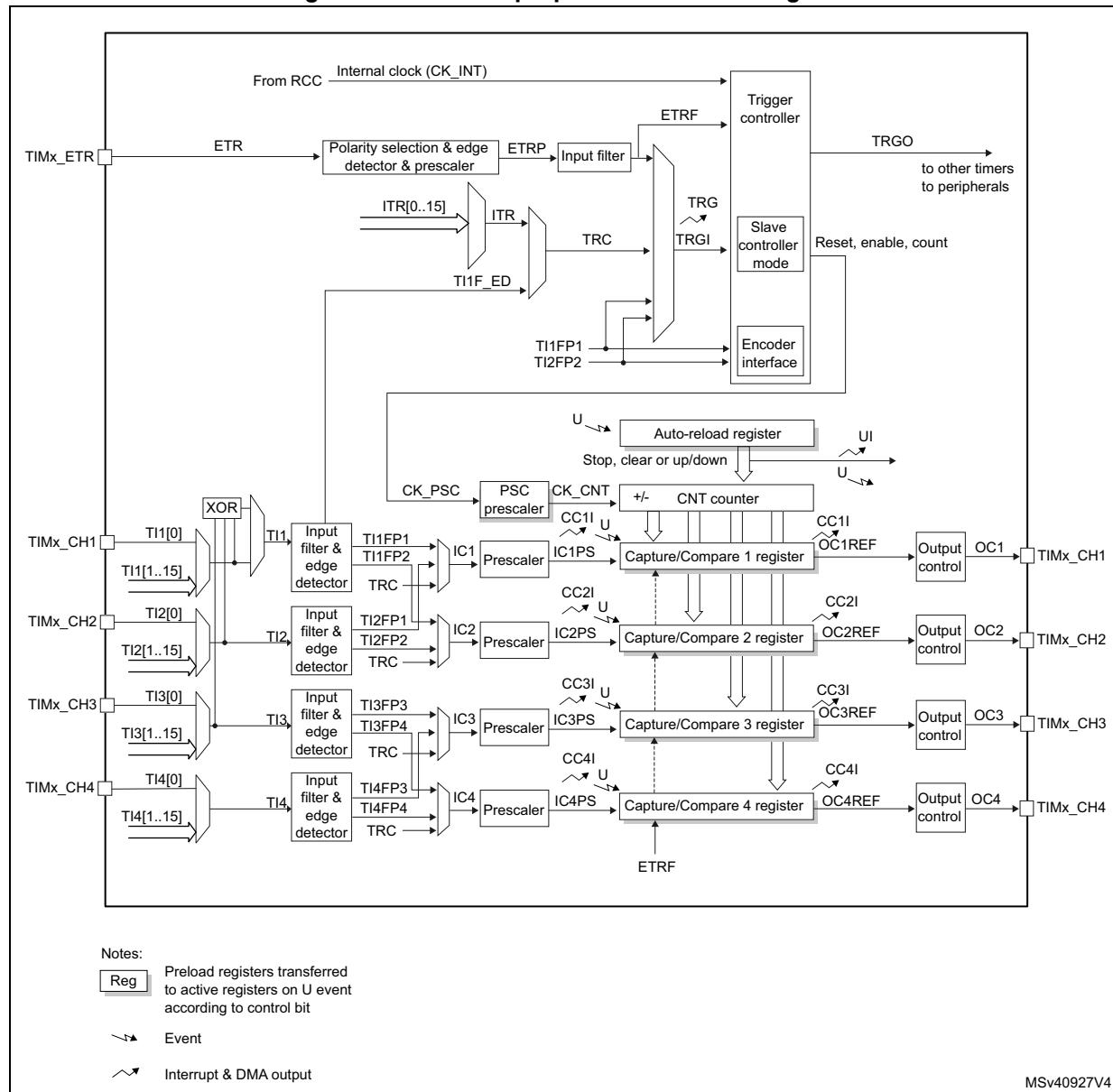
The timer may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

### 25.2 TIM2 main features

- 32-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 204. General-purpose timer block diagram



## 25.3 TIM2 functional description

### 25.3.1 Time-base unit

The main block of the programmable timer is a 32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx\_CNT)
- Prescaler Register (TIMx\_PSC)
- Auto-Reload Register (TIMx\_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT\_EN is set 1 clock cycle after CEN.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 205* and *Figure 206* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 205. Counter timing diagram with prescaler division change from 1 to 2

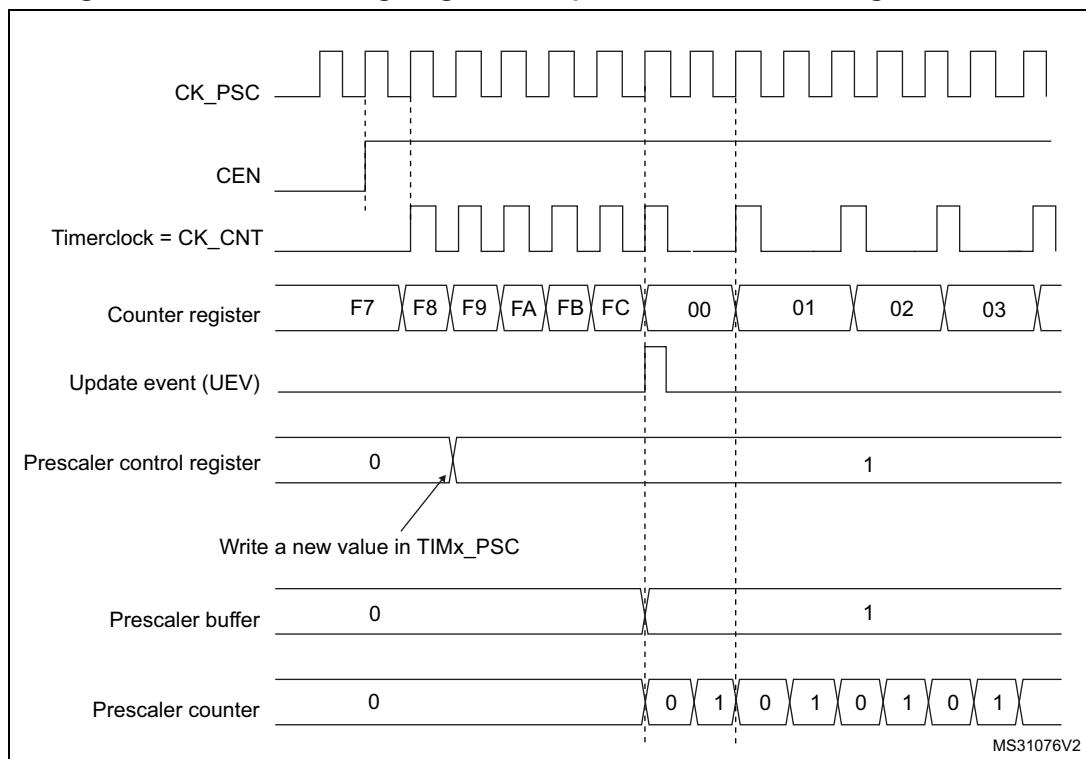
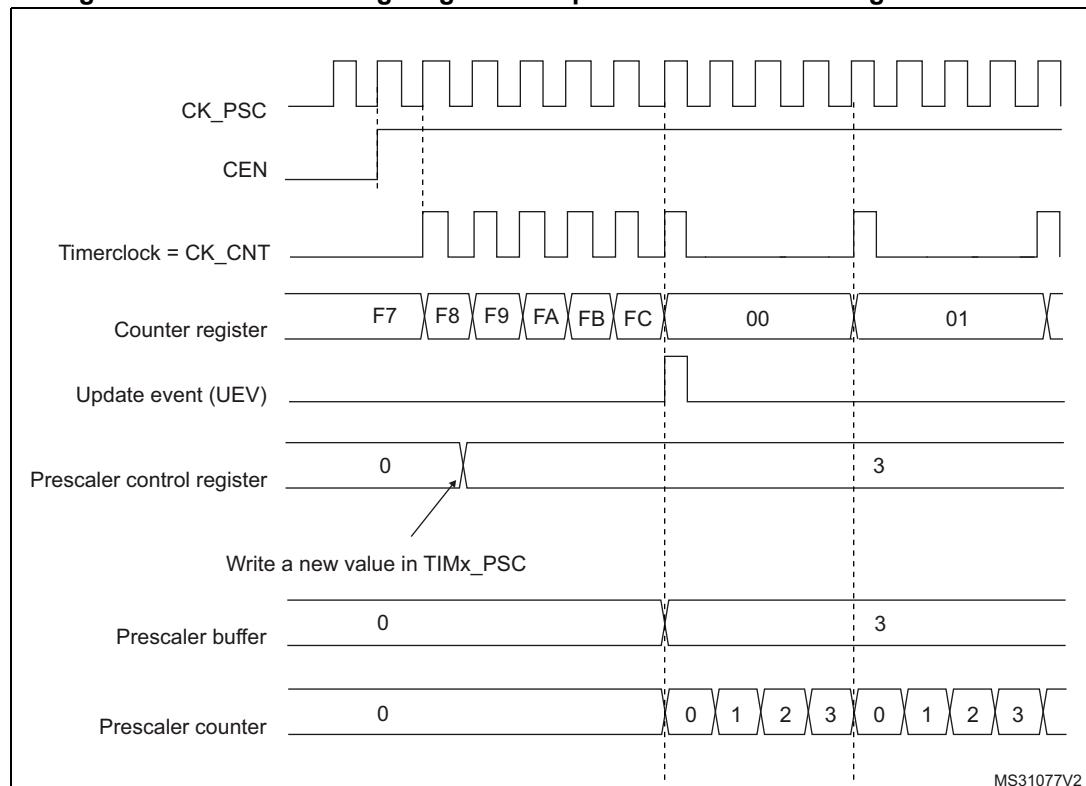


Figure 206. Counter timing diagram with prescaler division change from 1 to 4



### 25.3.2 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).

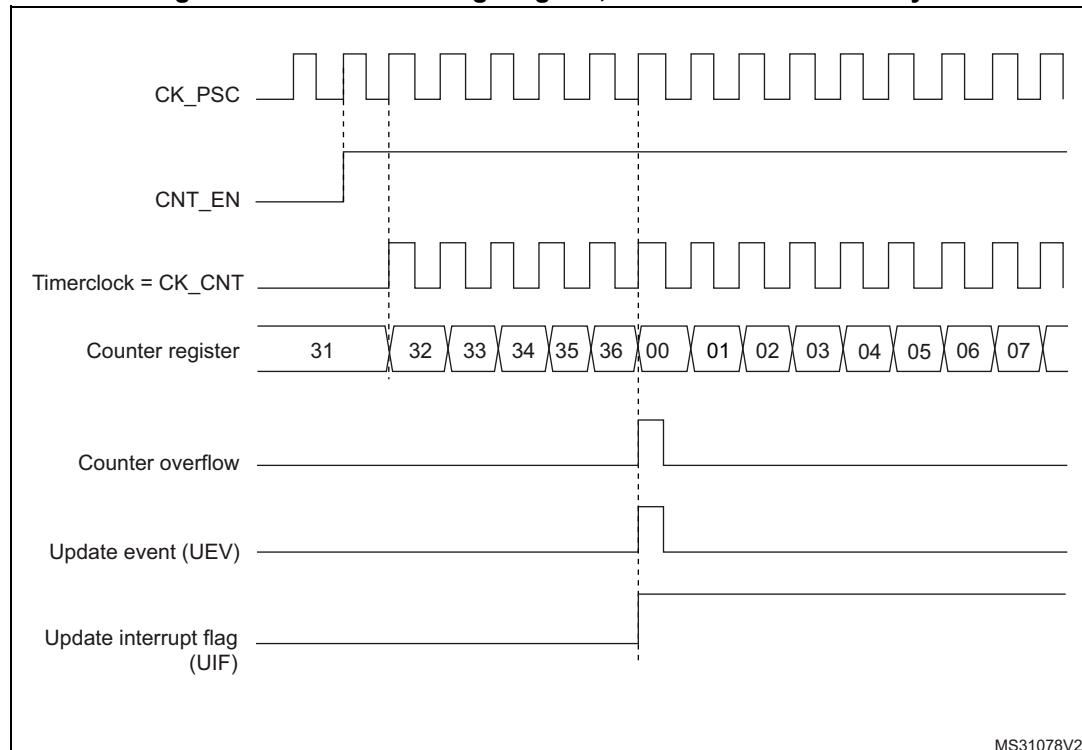
The UEV event can be disabled by software by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well 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 207. Counter timing diagram, internal clock divided by 1**



MS31078V2

Figure 208. Counter timing diagram, internal clock divided by 2

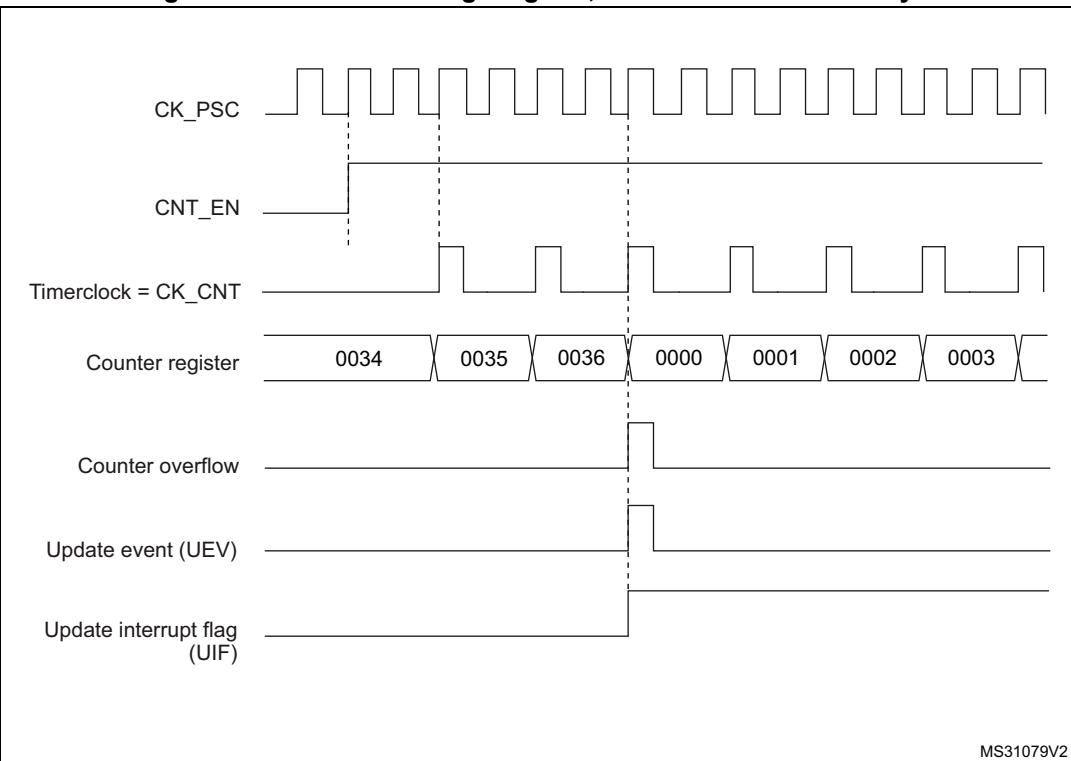
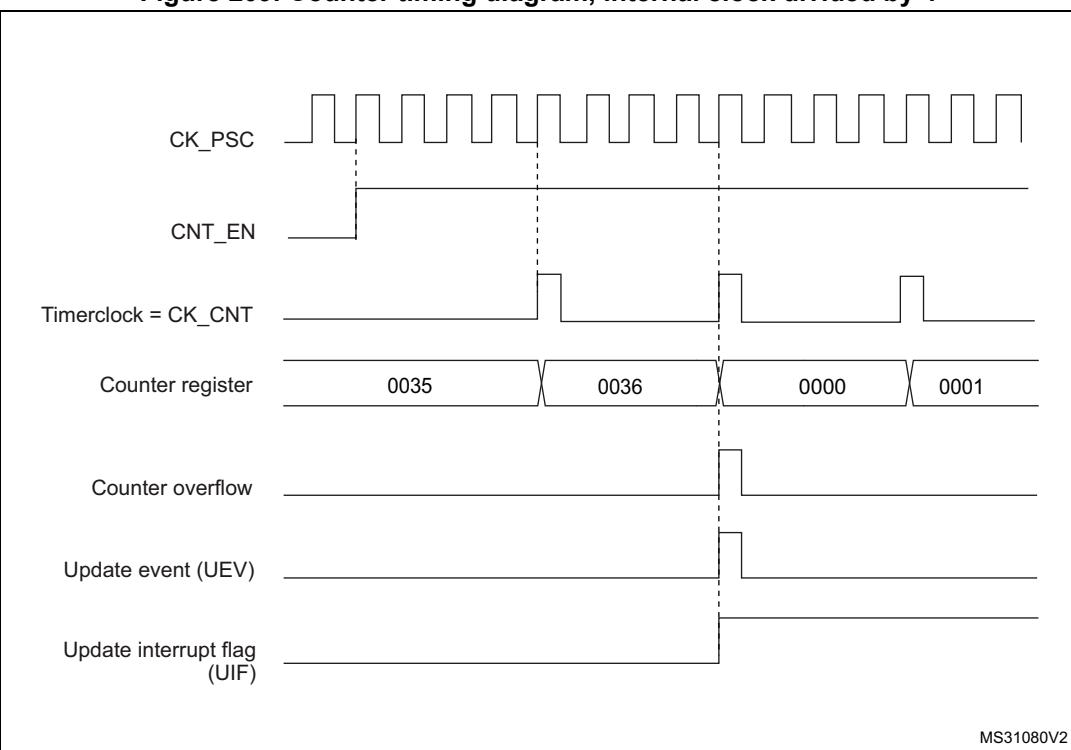
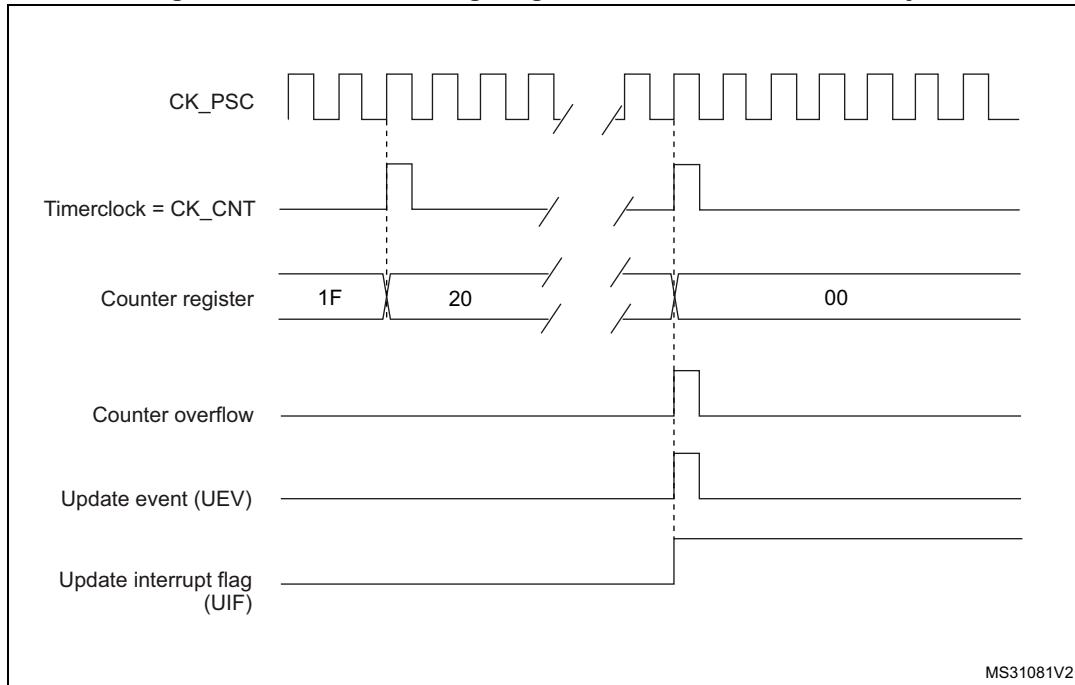


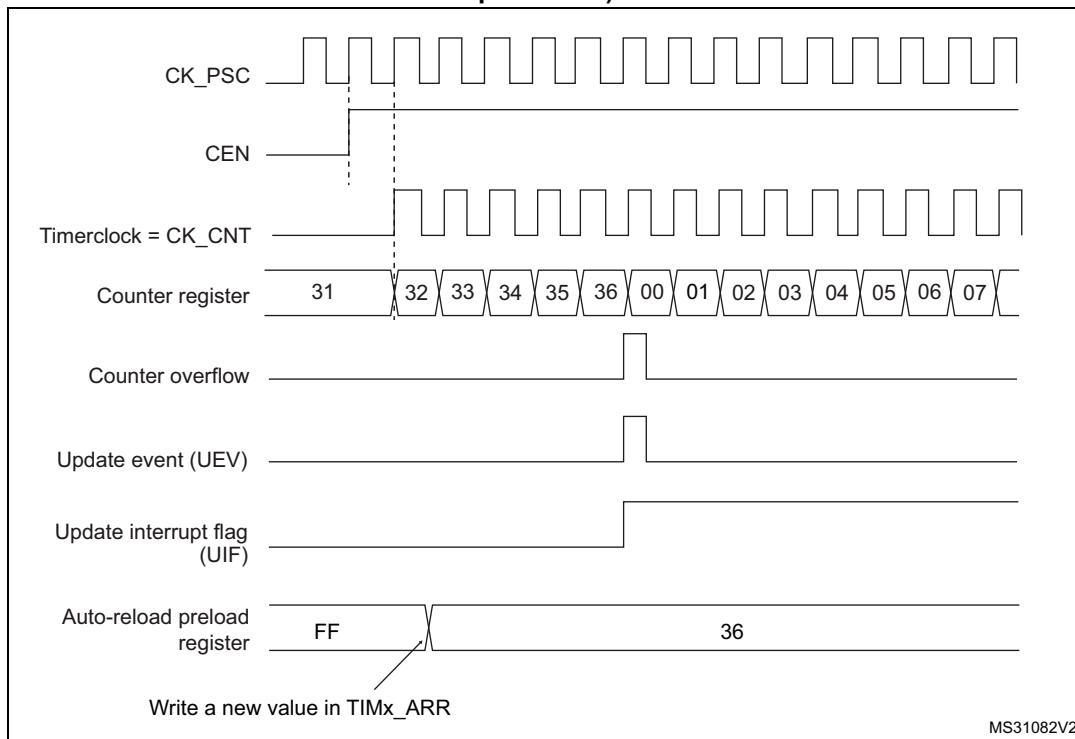
Figure 209. Counter timing diagram, internal clock divided by 4



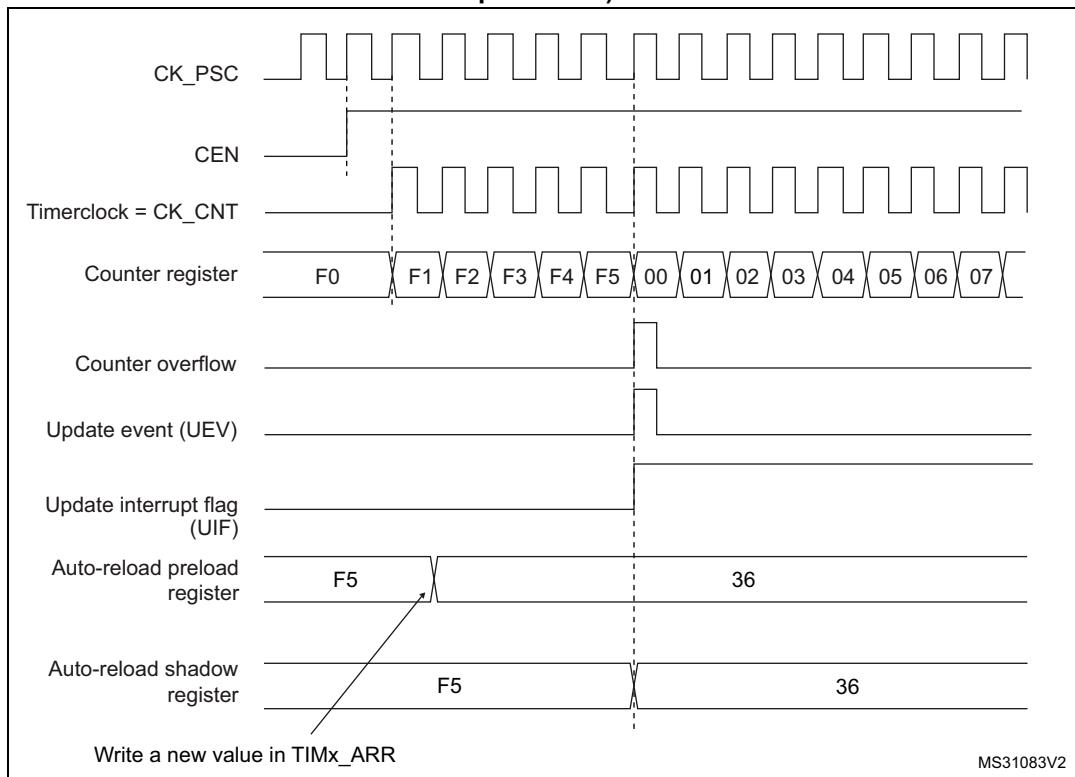
**Figure 210. Counter timing diagram, internal clock divided by N**



**Figure 211. Counter timing diagram, Update event when ARPE=0 (TIMx\_ARR not preloaded)**



**Figure 212. 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).

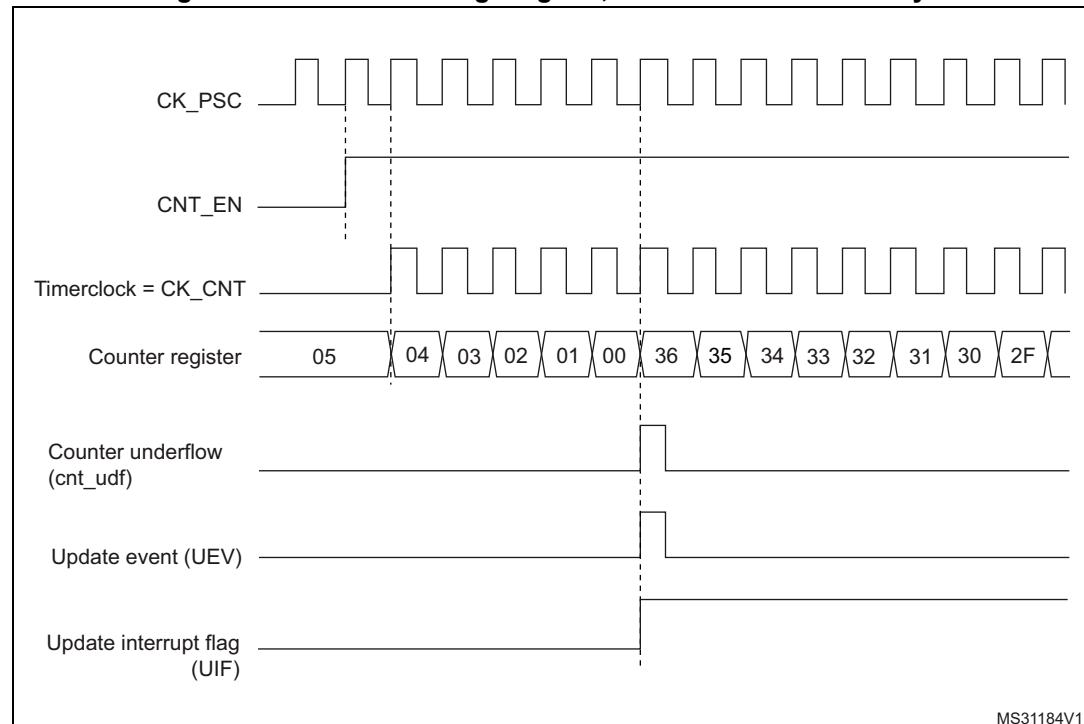
In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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  $\text{TIMx\_ARR}=0x36$ .

**Figure 213. Counter timing diagram, internal clock divided by 1**



**Figure 214. Counter timing diagram, internal clock divided by 2**

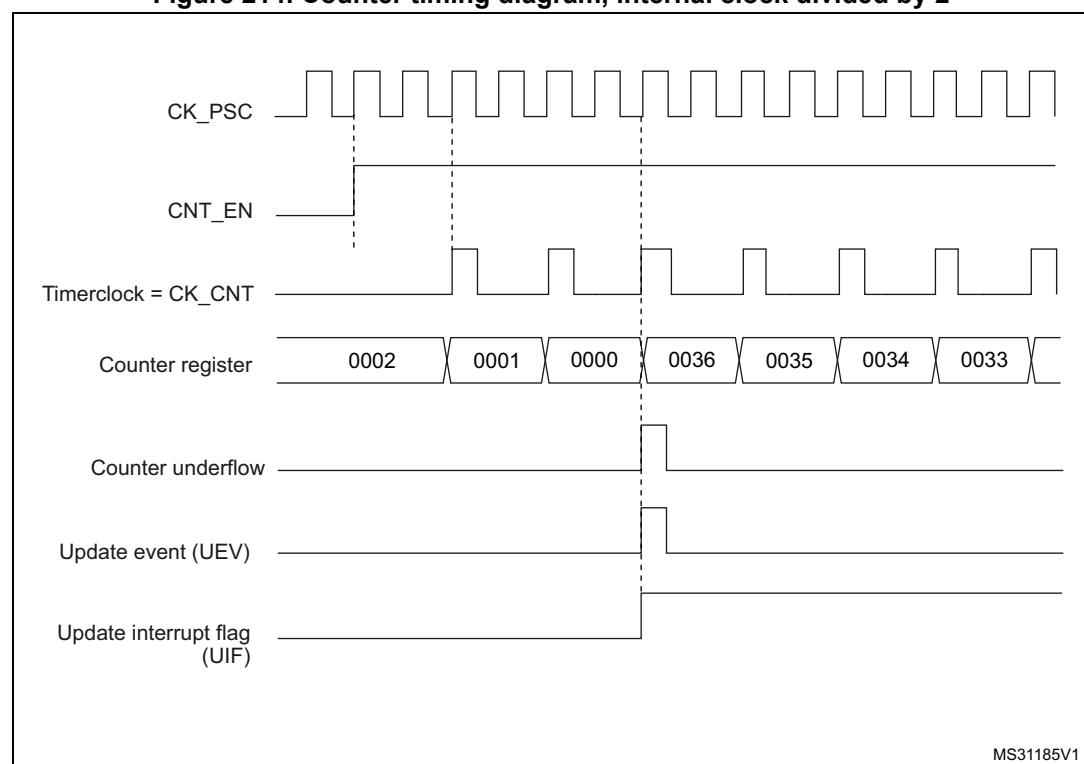


Figure 215. Counter timing diagram, internal clock divided by 4

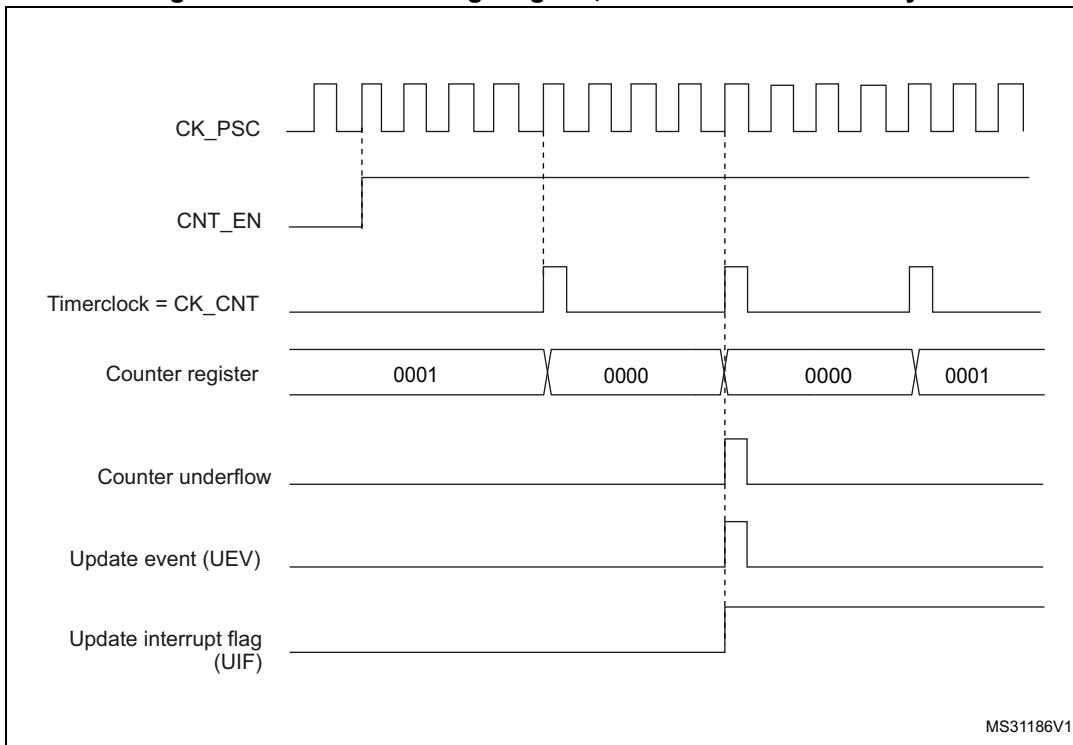
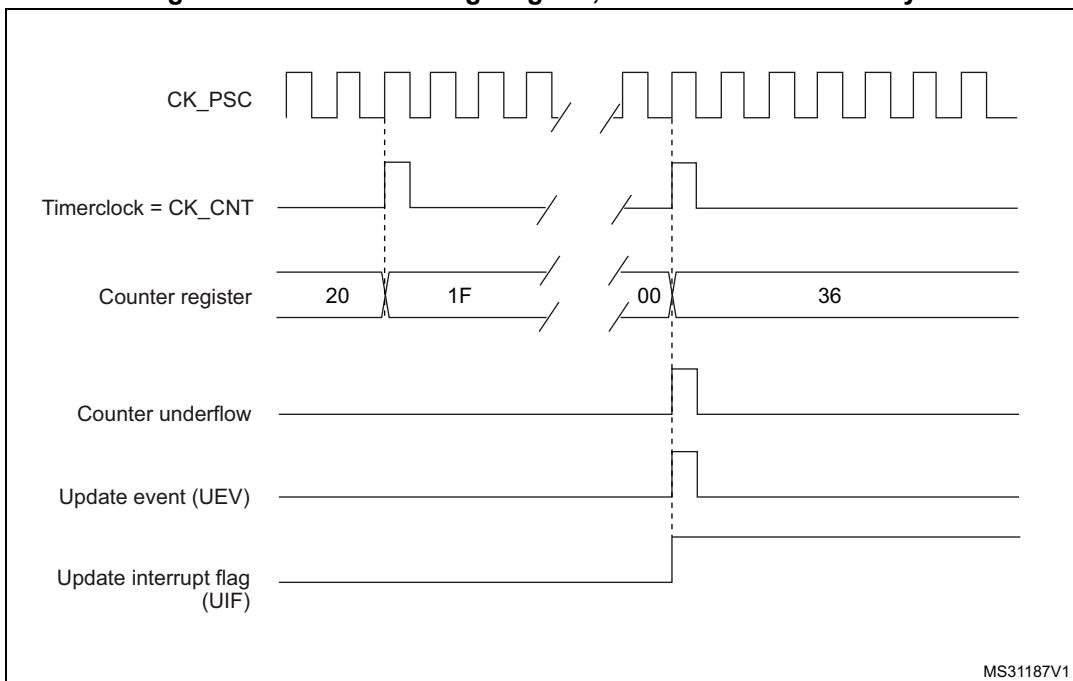
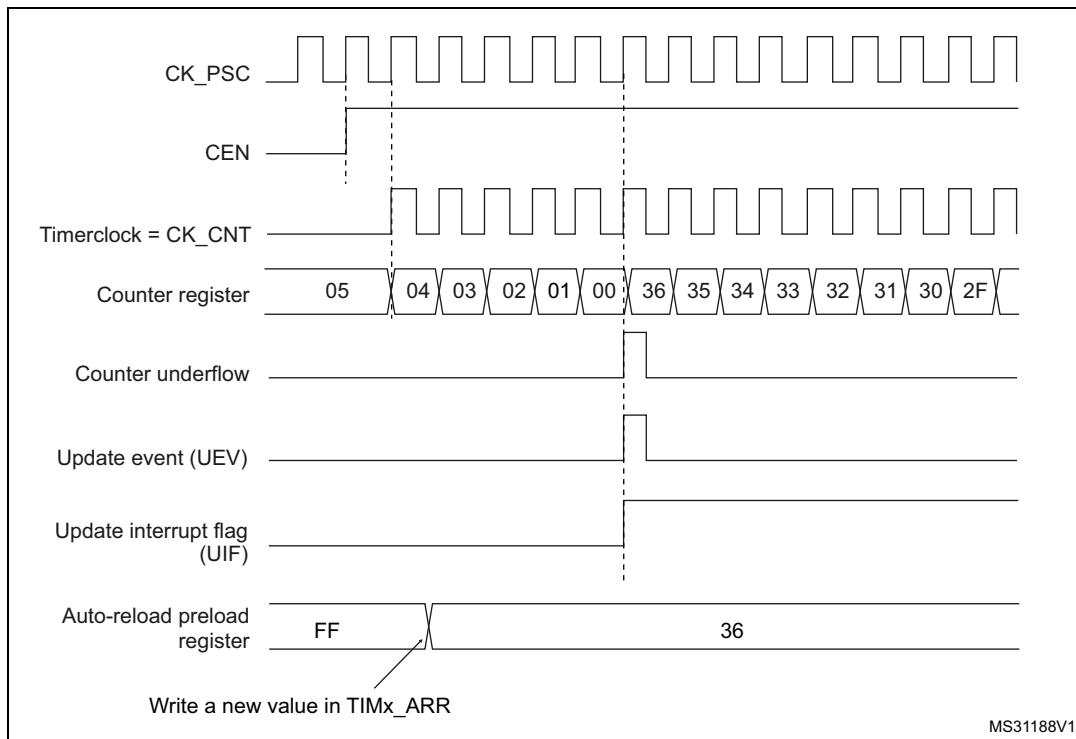


Figure 216. Counter timing diagram, internal clock divided by N



**Figure 217. Counter timing diagram, Update event when repetition counter is not used**



### Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx\_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or

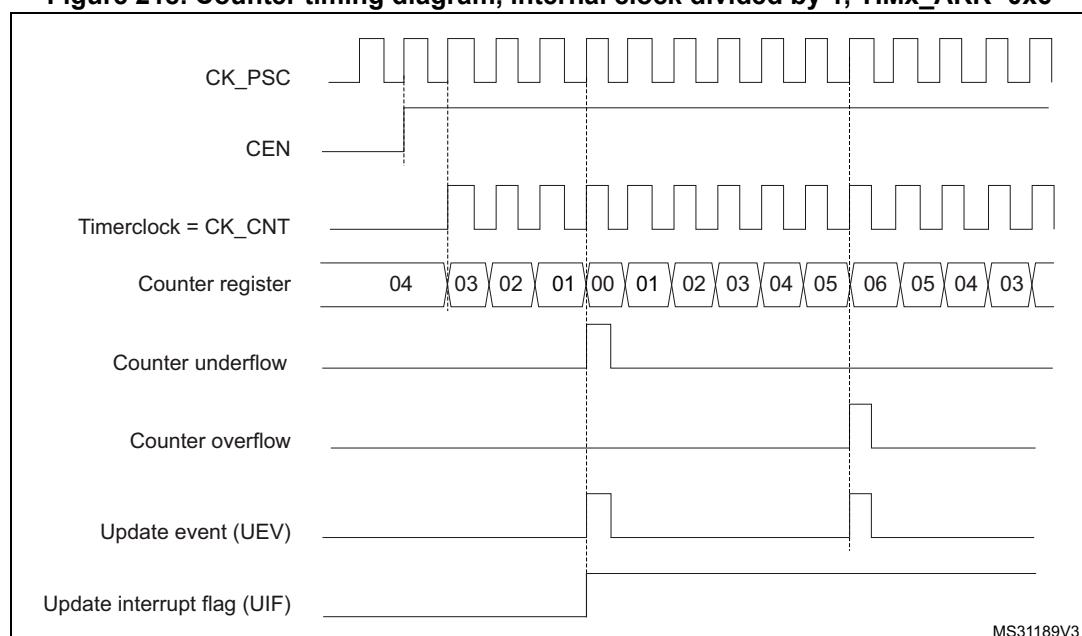
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

**Figure 218. 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 25.4.1: TIM2 control register 1 \(TIM2\\_CR1\) on page 806](#)).

Figure 219. Counter timing diagram, internal clock divided by 2

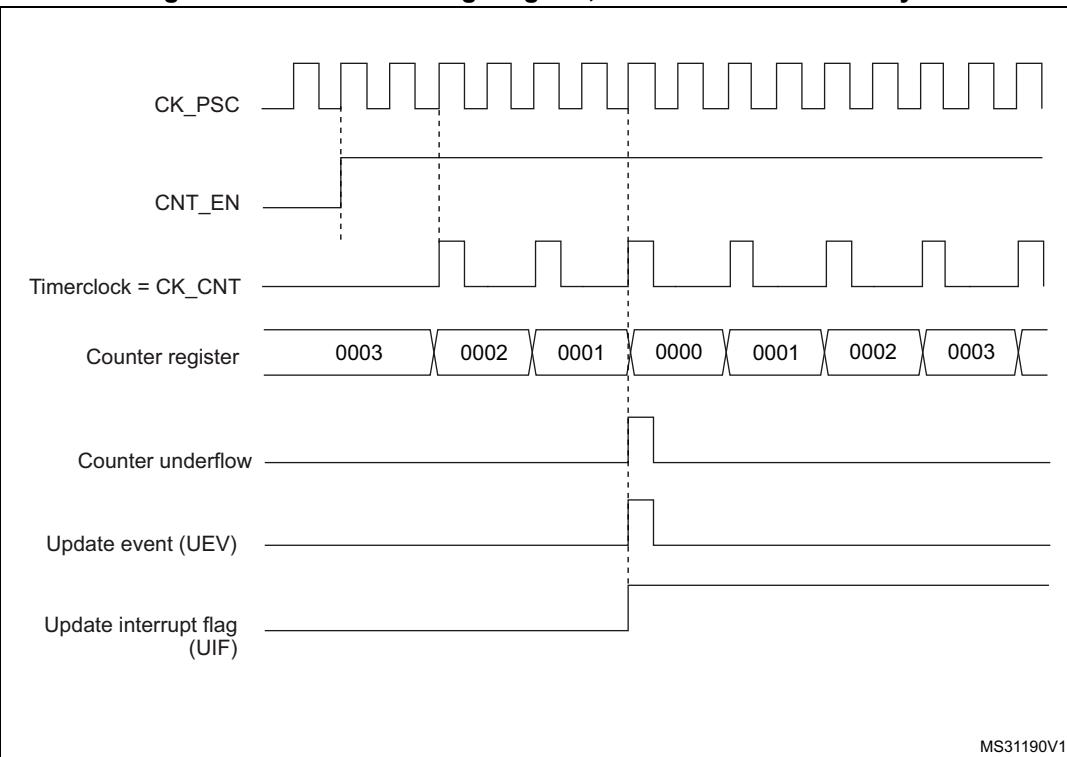
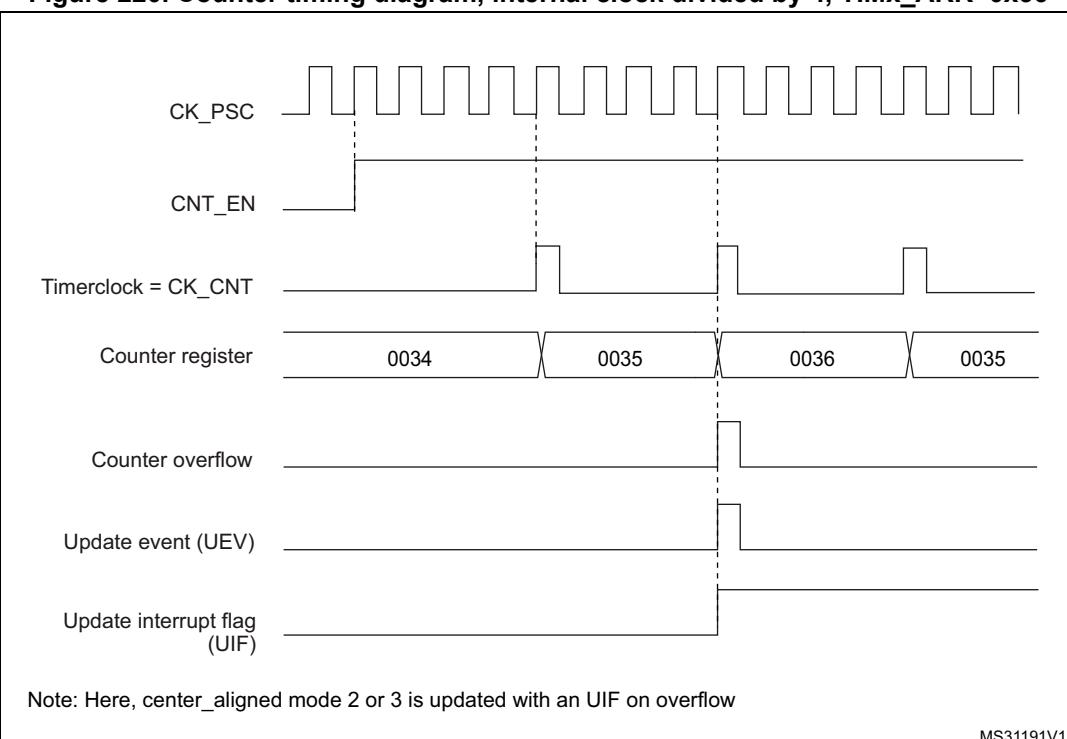


Figure 220. Counter timing diagram, internal clock divided by 4, TIMx\_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 221. Counter timing diagram, internal clock divided by N

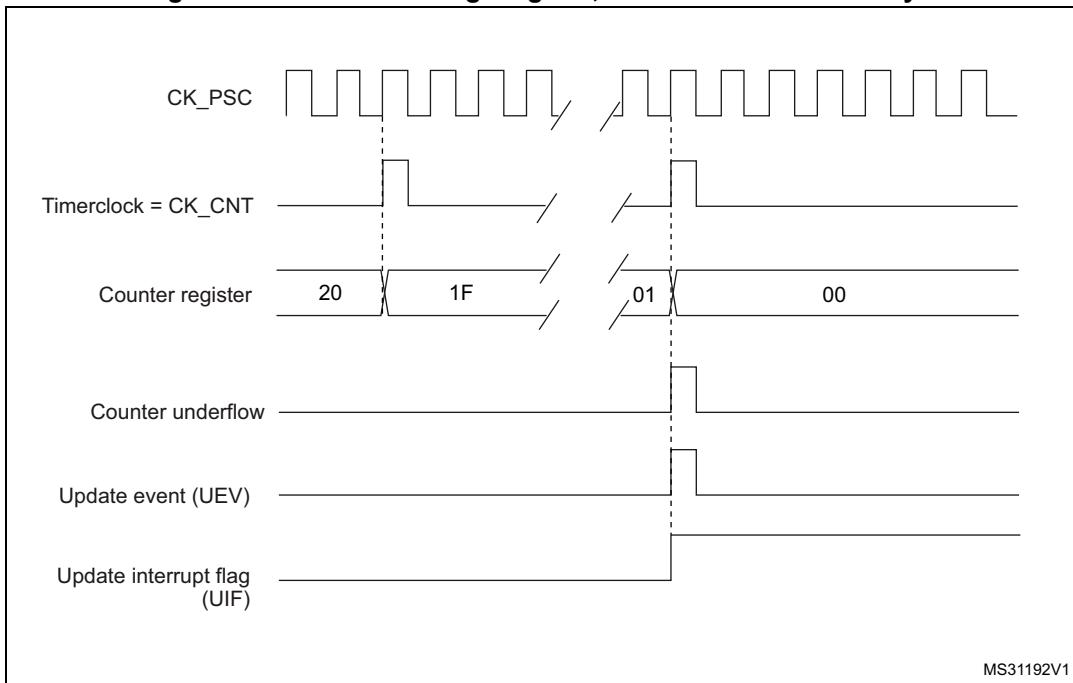


Figure 222. Counter timing diagram, Update event with ARPE=1 (counter underflow)

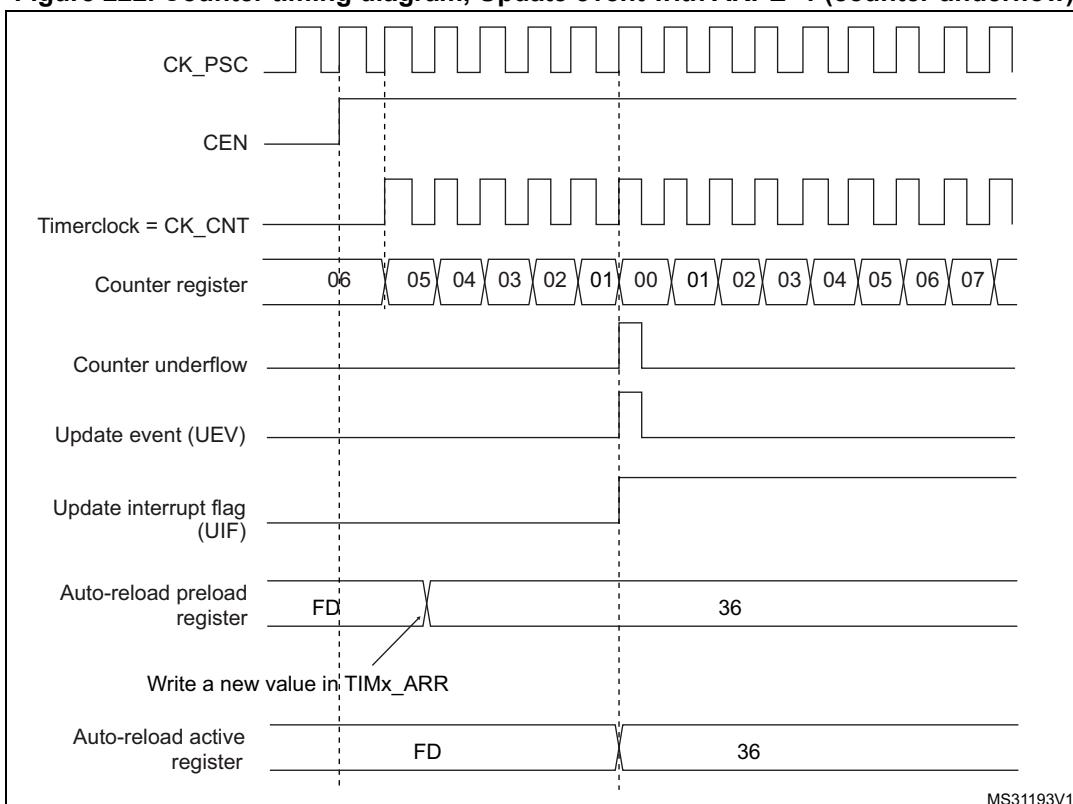
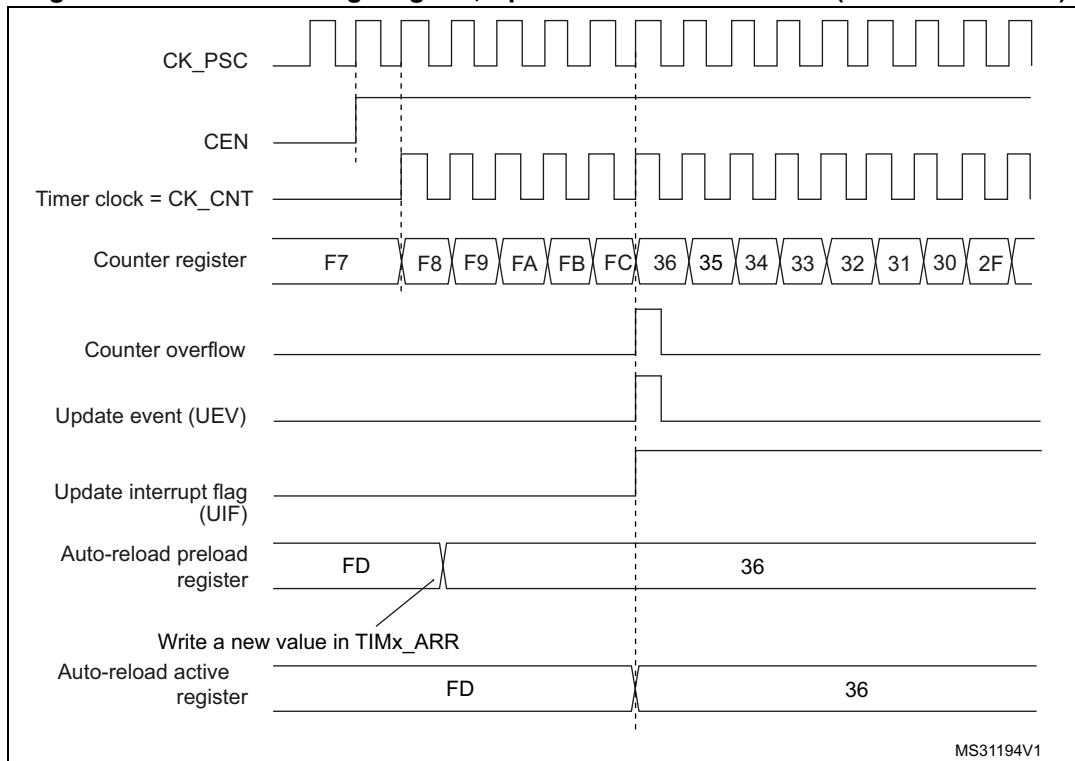


Figure 223. Counter timing diagram, Update event with ARPE=1 (counter overflow)



### 25.3.3 Clock selection

The counter clock can be provided by the following clock sources:

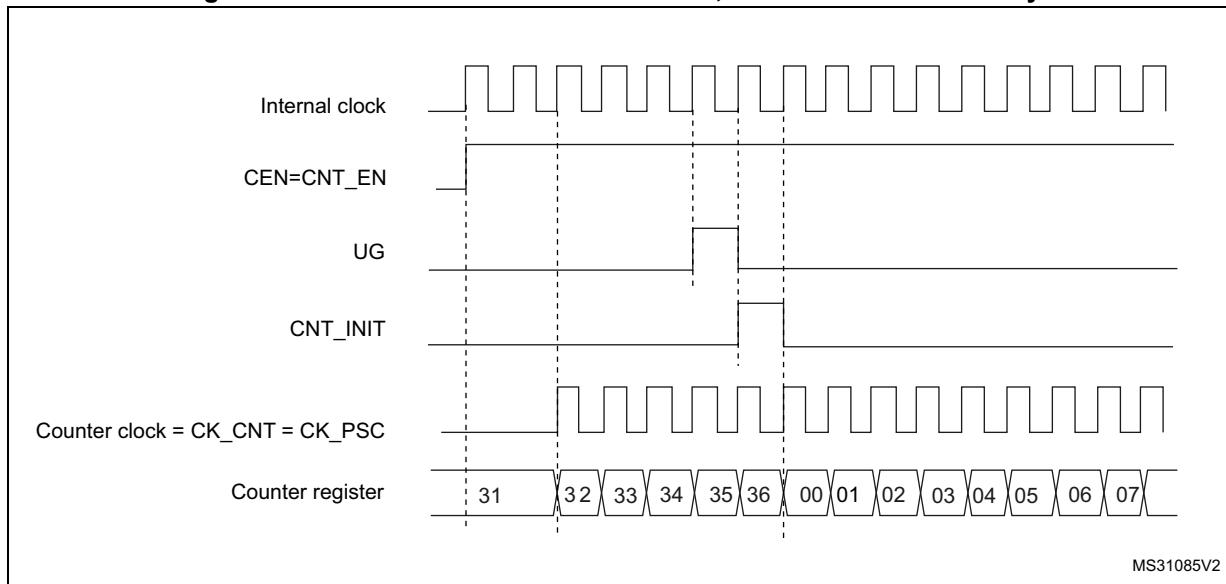
- Internal clock (CK\_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, Timer X can be configured to act as a prescaler for Timer Y. Refer to : [Using one timer as prescaler for another timer on page 801](#) 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 224](#) shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

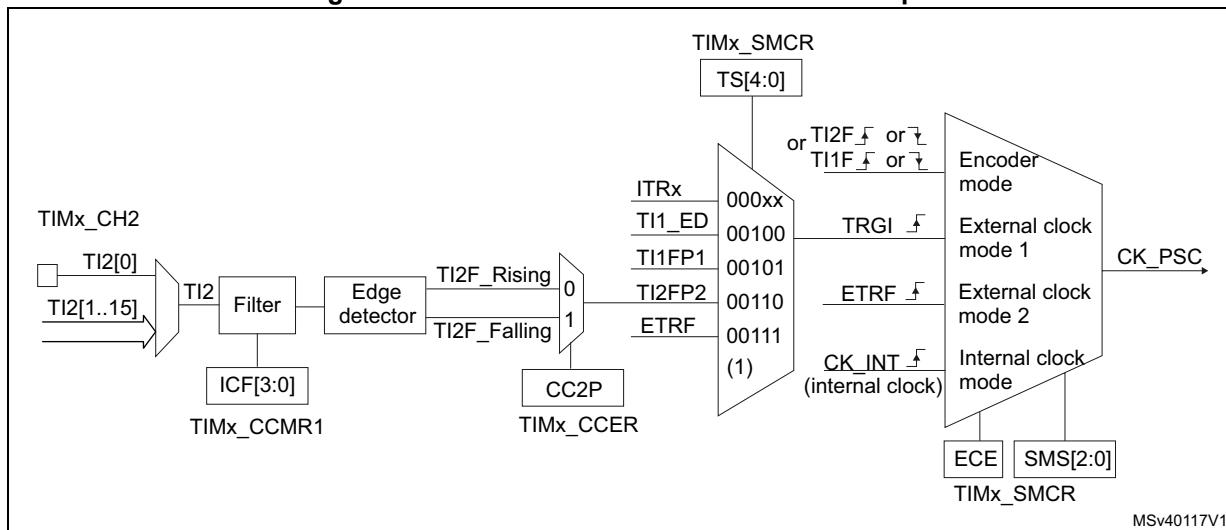
Figure 224. Control circuit in normal mode, internal clock divided by 1



## 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 225. TI2 external clock connection example**



- ## 1. Codes ranging from 01000 to 11111: ITRY.

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
  2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= '01 in the TIMx\_CCMR1 register.
  3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F=0000).

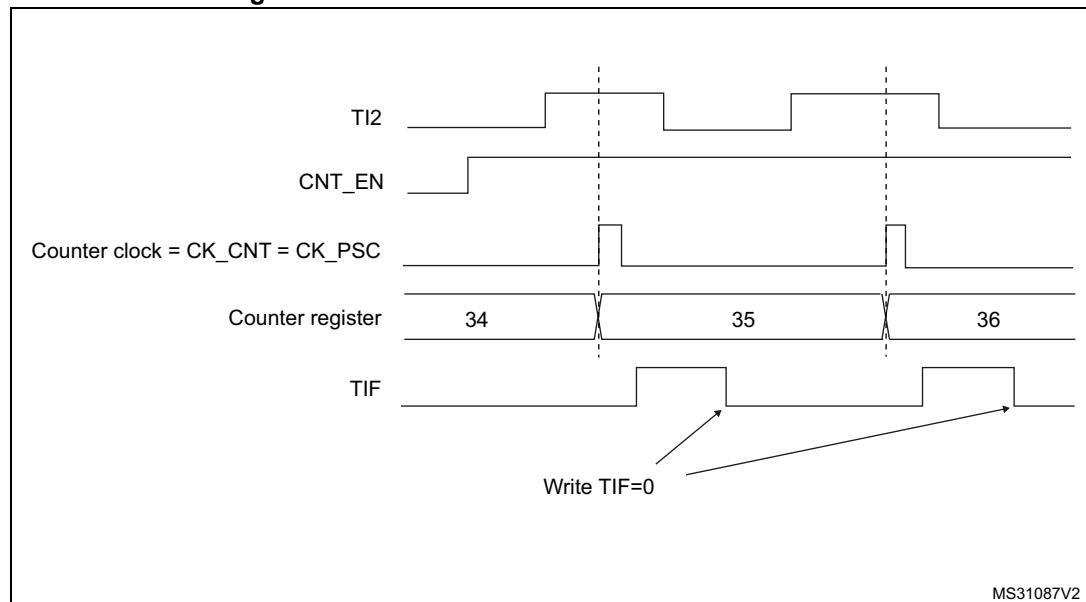
**Note:** The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 and CC2NP=0 in the TIMx\_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx\_SMCR register.
6. Select TI2 as the input source by writing TS=00110 in the TIMx\_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

**Figure 226. Control circuit in external clock mode 1**



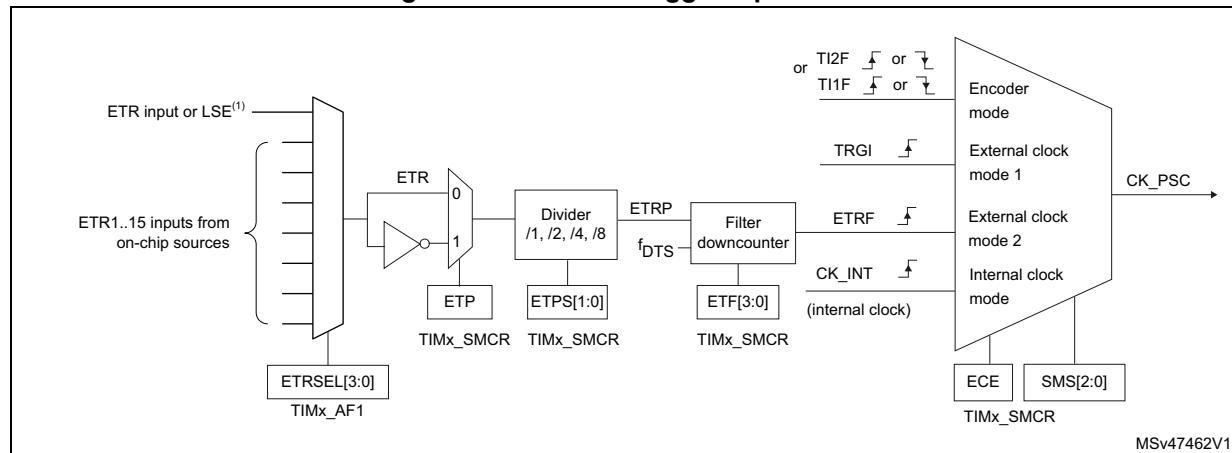
### External clock source mode 2

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 227](#) gives an overview of the external trigger input block.

Figure 227. External trigger input block



1. As per ETR\_RMP bit programming.

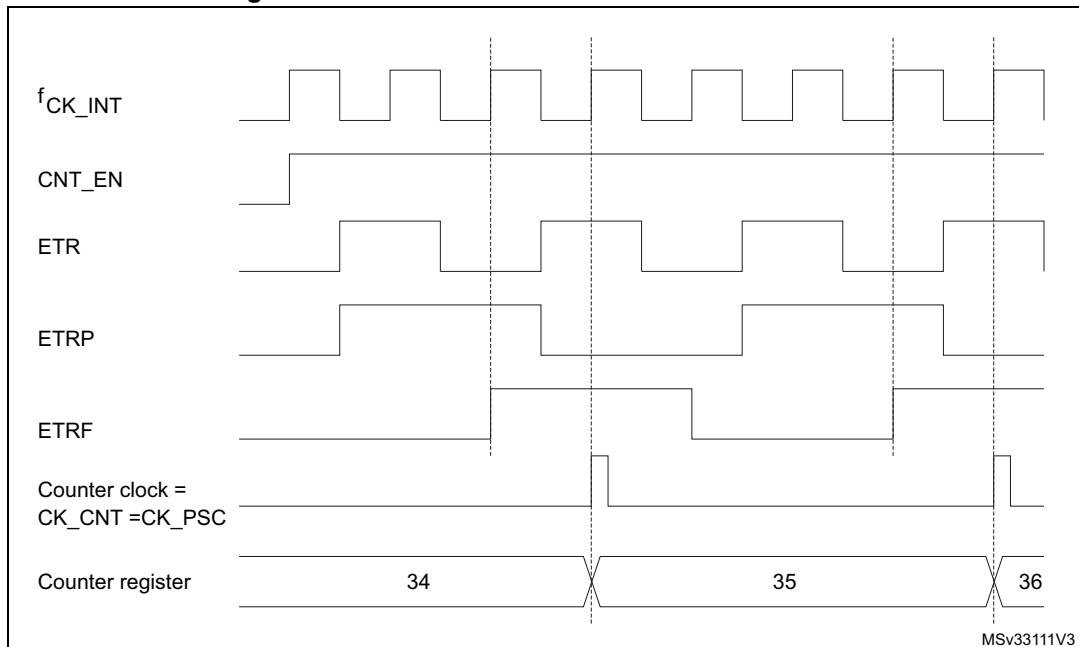
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Select the proper ETR source (internal or external) with the ETRSEL[3:0] bits in the TIMx\_AF1 register and the ETR\_RMP bit in the TIM2\_OR1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx\_SMCR register.
3. Set the prescaler by writing ETSPS[1:0]=01 in the TIMx\_SMCR register.
4. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx\_SMCR register.
5. Enable external clock mode 2 by writing ECE=1 in the TIMx\_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most 1/4 of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETSPS prescaler setting.

Figure 228. Control circuit in external clock mode 2



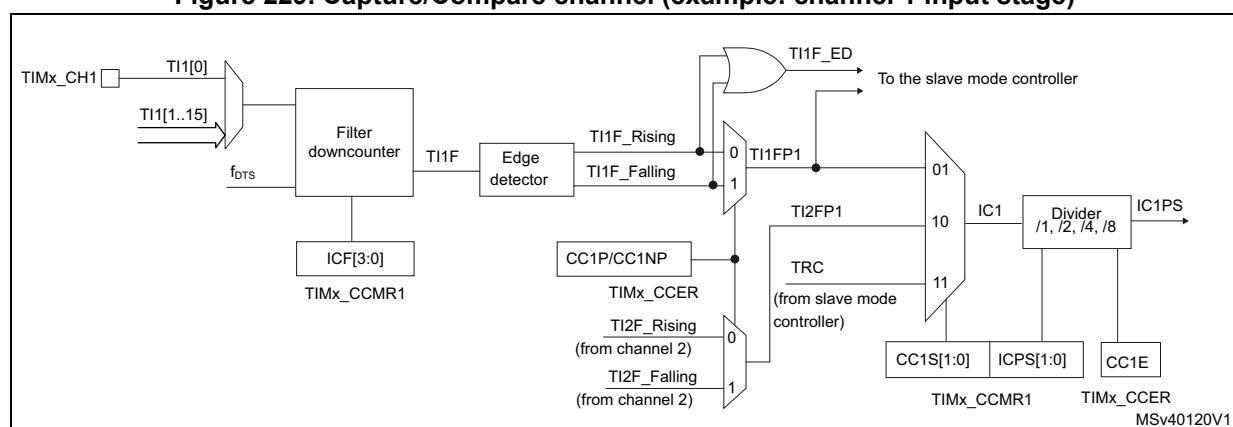
### 25.3.4 Capture/Compare channels

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  $Tlx$  input to generate a filtered signal  $Tlx_F$ . Then, an edge detector with polarity selection generates a signal ( $Tlx_{FPx}$ ) 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 229. 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 230. Capture/Compare channel 1 main circuit

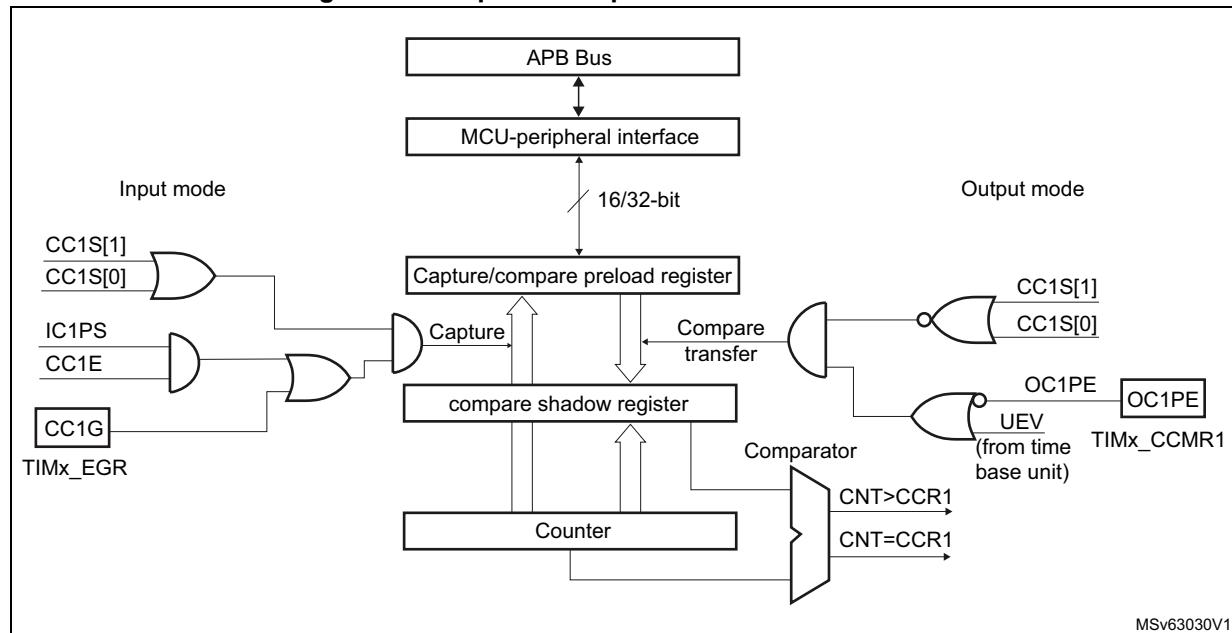
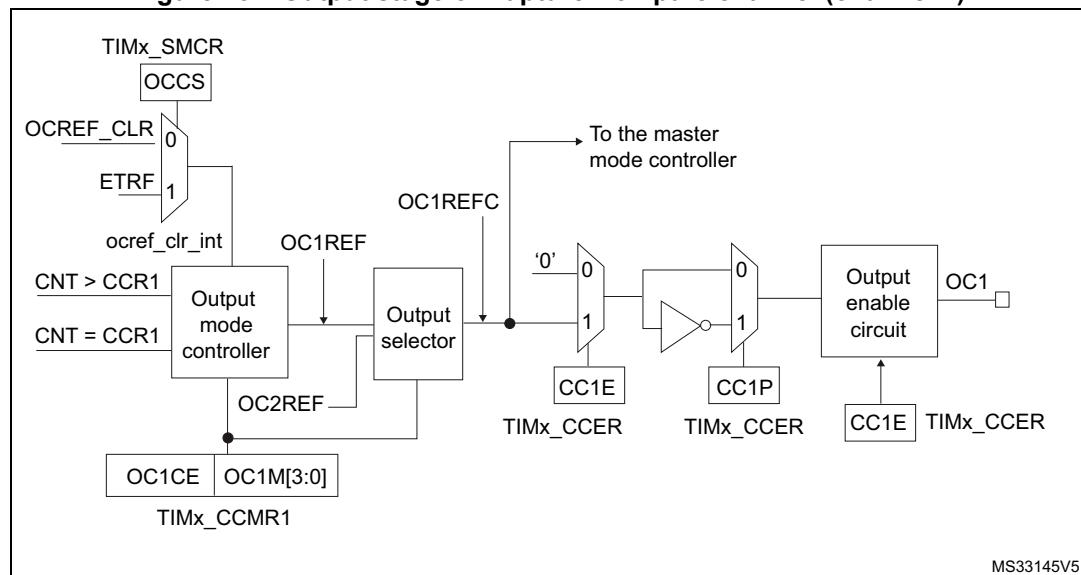


Figure 231. Output stage of Capture/Compare channel (channel 1)



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.

### 25.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx\_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx\_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx\_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx\_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx\_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at  $f_{DTS}$  frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx\_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx\_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx\_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

**Note:** *IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx\_EGR register.*

### 25.3.6 PWM input mode

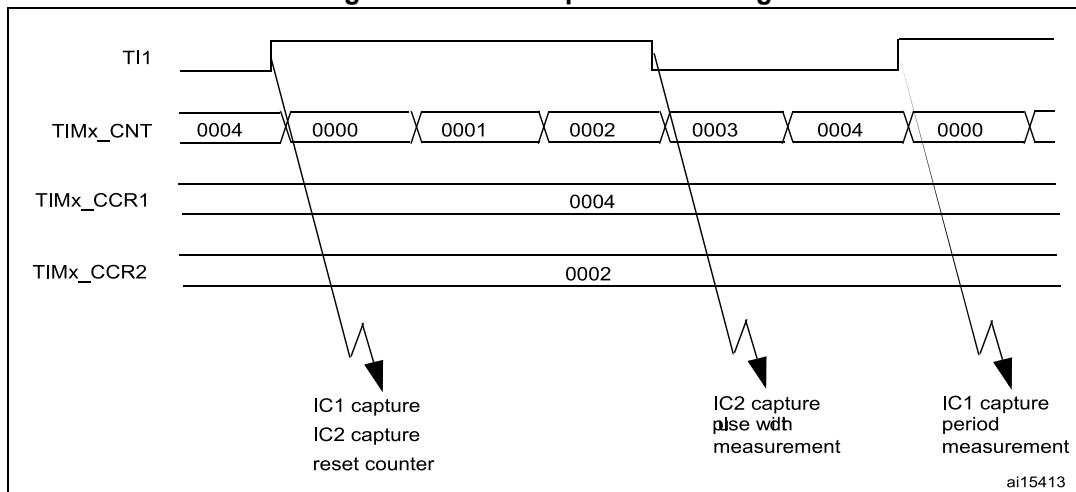
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx\_CCR1 register) and the duty cycle (in TIMx\_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK\_INT frequency and prescaler value):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input for TIMx\_CCR1: write the CC1S bits to 01 in the TIMx\_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx\_CCR1 and counter clear): write the CC1P to '0' and the CC1NP bit to '0' (active on rising edge).
4. Select the active input for TIMx\_CCR2: write the CC2S bits to 10 in the TIMx\_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx\_CCR2): write the CC2P bit to '1' and the CC2NP bit to '0' (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx\_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx\_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx\_CCER register.

Figure 232. PWM input mode timing



1. The PWM input mode can be used only with the TIMx\_CH1/TIMx\_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

### 25.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

### 25.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCXM=000), be set active (OCXM=001), be set inactive (OCXM=010) or can toggle (OCXM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx\_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register.

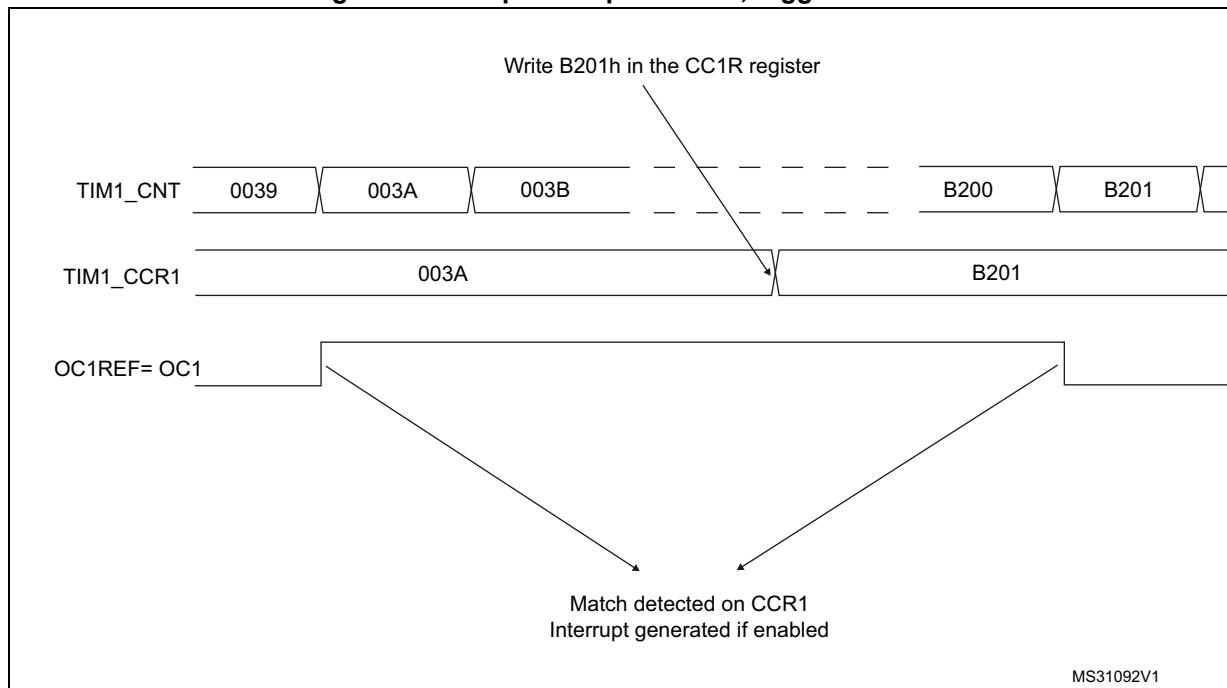
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

#### Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, one must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 233](#).

**Figure 233. Output compare mode, toggle on OC1**



### 25.3.9 PWM mode

Pulse width modulation mode permits to generate a signal with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or '111 (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx\_CCER register. Refer to the TIMx\_CCERx register description for more details.

In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether TIMx\_CCRx  $\leq$  TIMx\_CNT or TIMx\_CNT  $\leq$  TIMx\_CCRx (depending on the direction of the counter). However, to comply with the OCREF\_CLR functionality (OCREF can be

cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison or
- When the output compare mode (OCxM bits in TIMx\_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM='000) to one of the PWM modes (OCxM='110 or '111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx\_CR1 register.

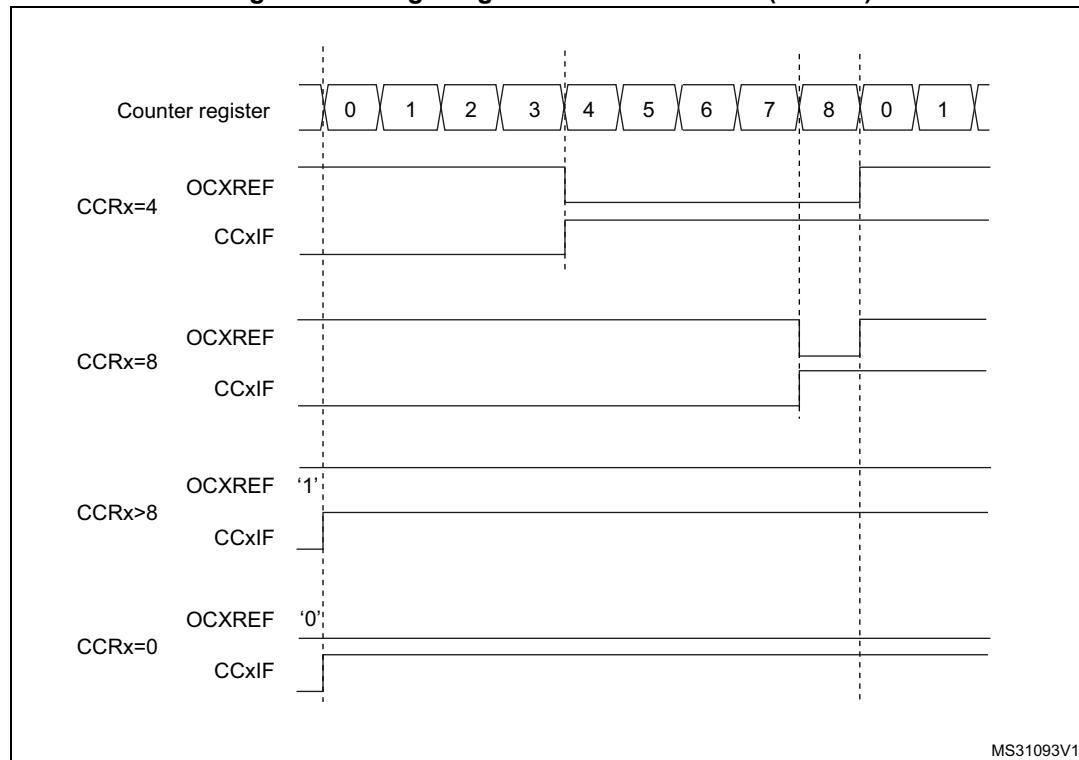
### PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low. Refer to [Upcounting mode on page 766](#).

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 234](#) shows some edge-aligned PWM waveforms in an example where TIMx\_ARR=8.

**Figure 234. Edge-aligned PWM waveforms (ARR=8)**



MS31093V1

## Downcounting configuration

Downcounting is active when DIR bit in TIMx\_CR1 register is high. Refer to [Downcounting mode on page 769](#).

In PWM mode 1, the reference signal ocxref is low as long as TIMx\_CNT>TIMx\_CCRx else it becomes high. If the compare value in TIMx\_CCRx is greater than the auto-reload value in TIMx\_ARR, then ocxref is held at 100%. PWM is not possible in this mode.

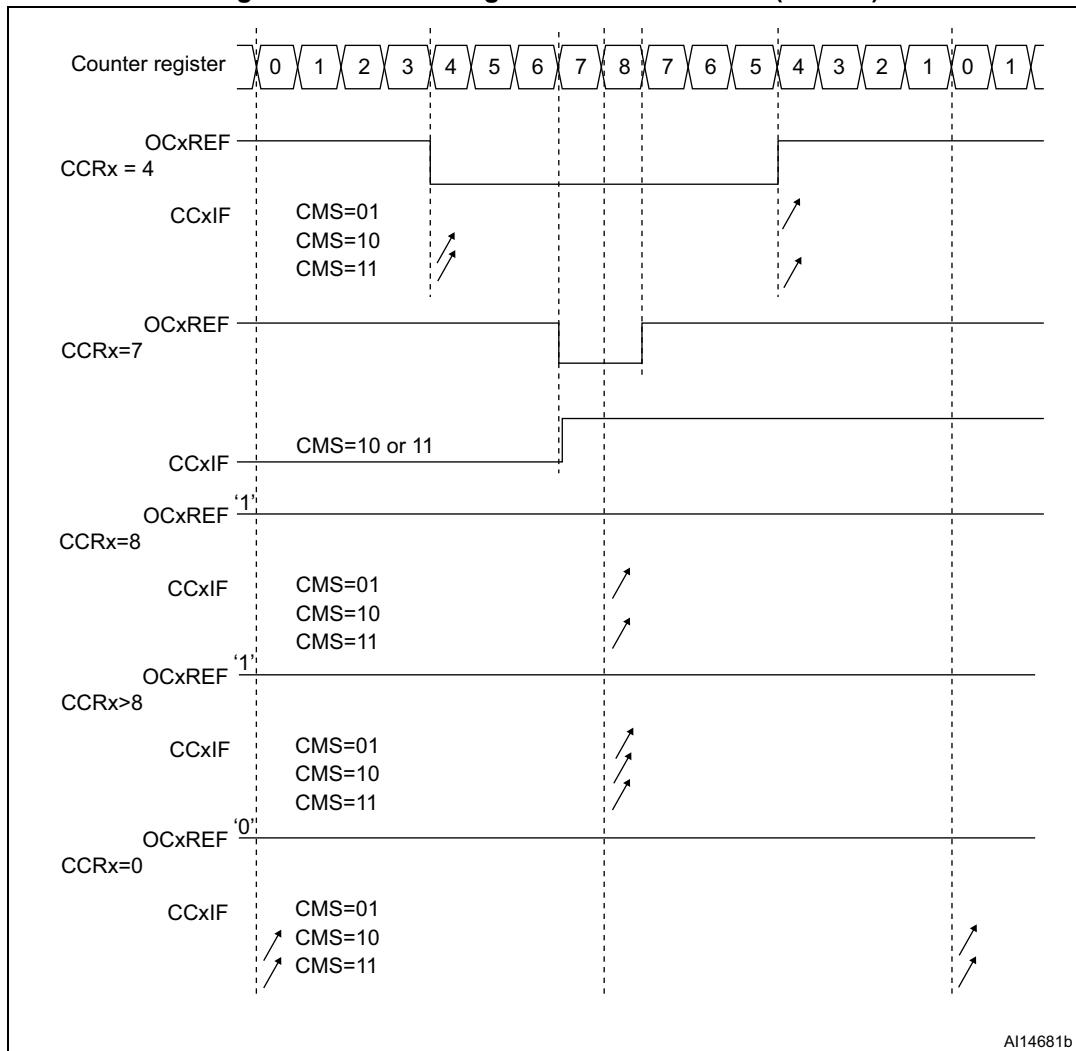
## PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are different from '00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx\_CR1 register is updated by hardware and must not be changed by software. Refer to [Center-aligned mode \(up/down counting\) on page 772](#).

*Figure 235* 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 235. Center-aligned PWM waveforms (ARR=8)



AI14681b

## Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx\_CNT > TIMx\_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx\_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx\_EGR register) just before starting the counter and not to write the counter while it is running.

### 25.3.10 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx\_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx\_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx\_CCR1 and TIMx\_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx\_CCR3 and TIMx\_CCR4

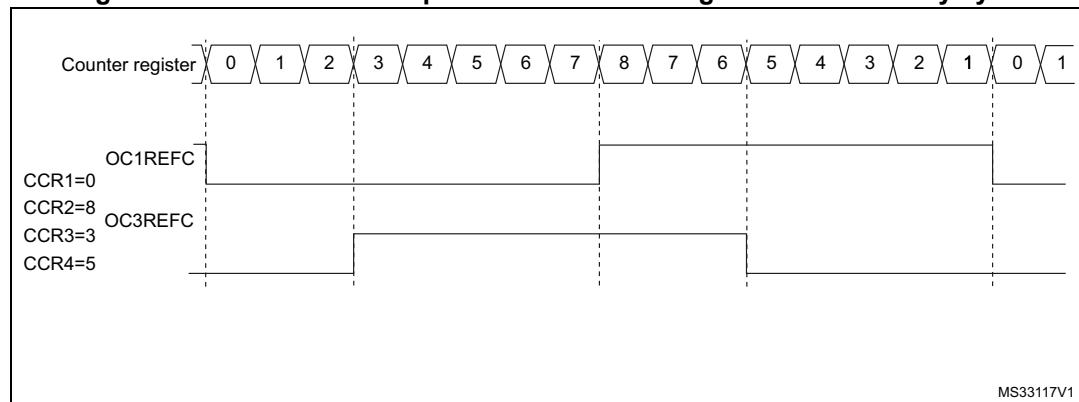
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1110' (Asymmetric PWM mode 1) or '1111' (Asymmetric PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

**Note:** *The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.*

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 2.

[Figure 236](#) shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1).

**Figure 236. Generation of 2 phase-shifted PWM signals with 50% duty cycle**



### 25.3.11 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx\_ARR register, the duty cycle and delay are determined by the two TIMx\_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx\_CCR1 and TIMx\_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx\_CCR3 and TIMx\_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

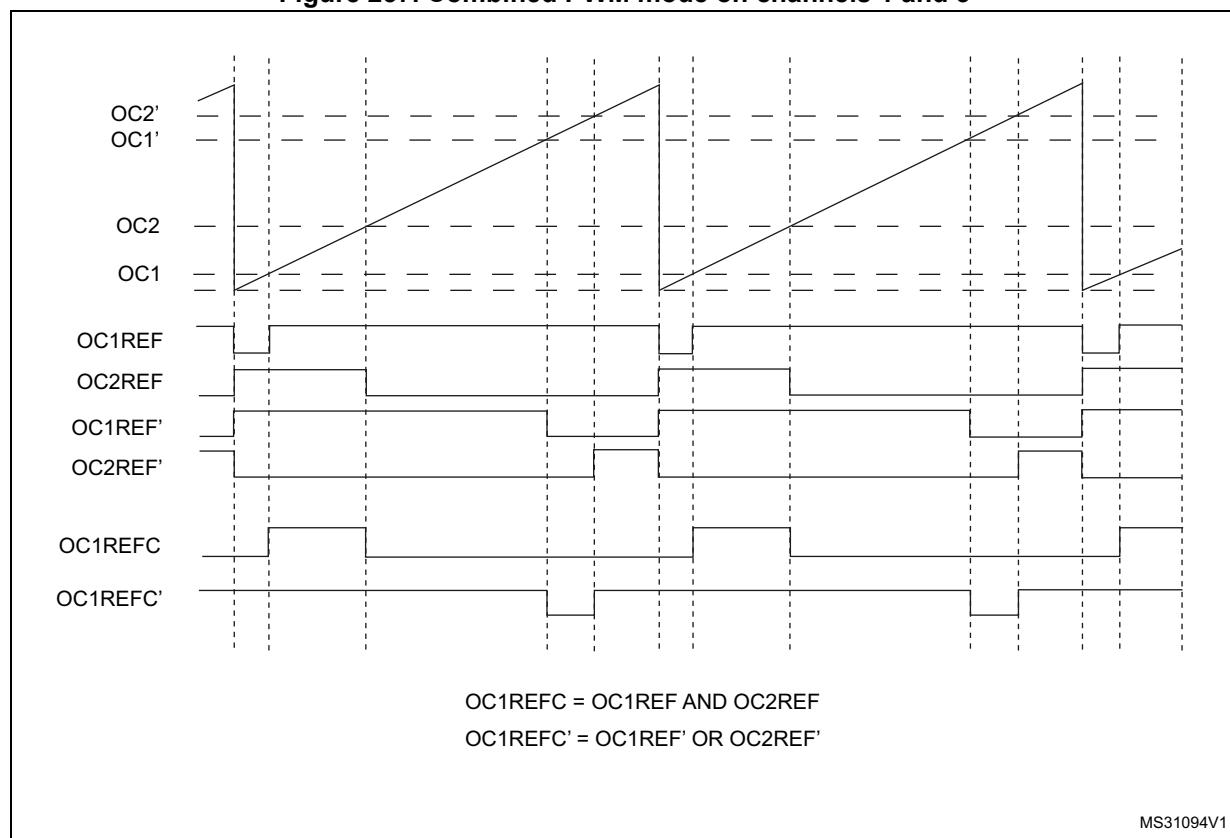
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

**Note:** *The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.*

*Figure 237* shows an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1

**Figure 237. Combined PWM mode on channels 1 and 3**



### 25.3.12 Clearing the OCxREF signal on an external event

The OCxREF signal of a given channel can be cleared when a high level is applied on the `ocref_clr_int` input (OCxCE enable bit in the corresponding `TIMx_CCMRx` register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

`OCREF_CLR_INPUT` can be selected between the `OCREF_CLR` input and `ETRF` (`ETR` after the filter) by configuring the `OCCS` bit in the `TIMx_SMCR` register.

The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx\_CCMRx register). OCxREF remains low until the next update event (UEV) occurs.

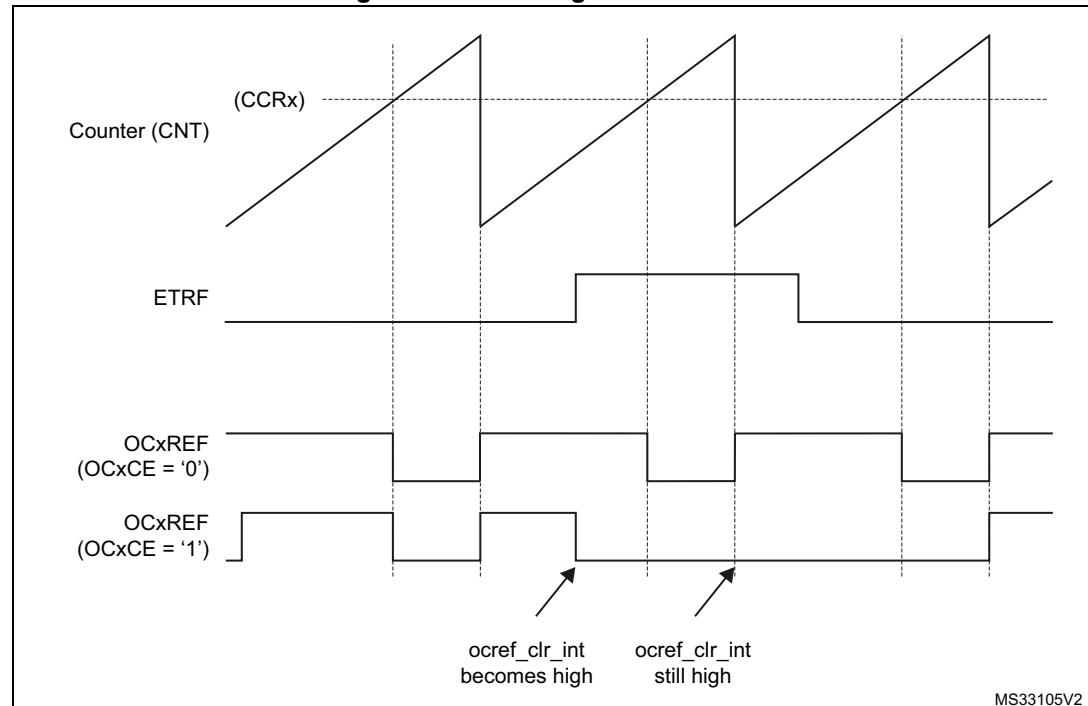
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx\_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1\_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

*Figure 238* 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 238. Clearing TIMx OCxREF**



*Note:*

*In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.*

### 25.3.13 One-pulse mode

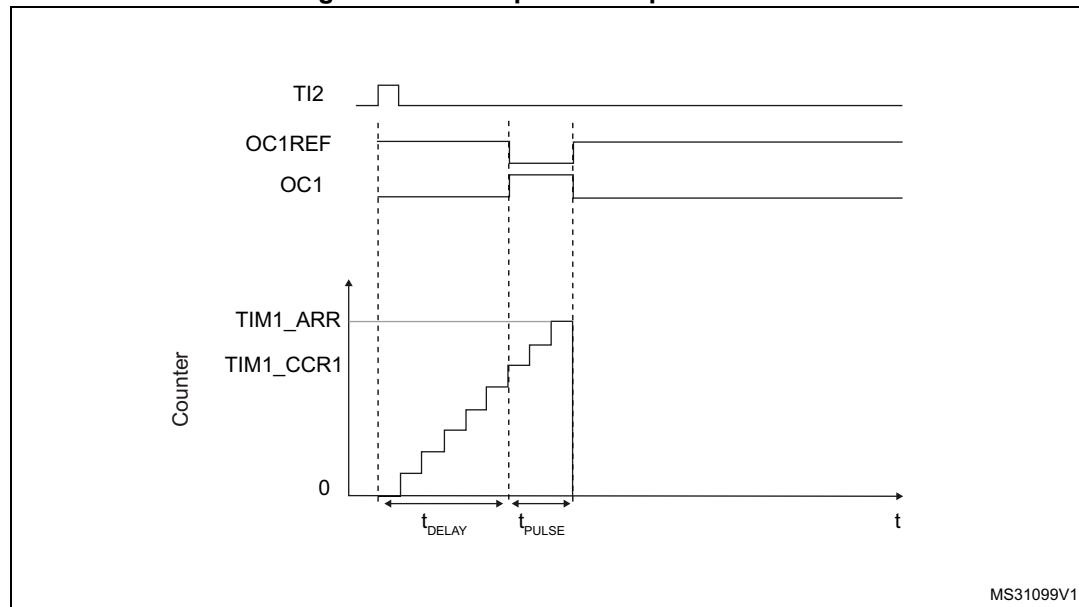
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $CNT < CCRx \leq ARR$  (in particular,  $0 < CCRx$ ),

Figure 239. Example of one-pulse mode.



For example one may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx\_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx\_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx\_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110 in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{DELAY}$  is defined by the value written in the `TIMx_CCR1` register.
- The  $t_{PULSE}$  is defined by the difference between the auto-reload value and the compare value (`TIMx_ARR` - `TIMx_CCR1`).
- Let's say one want to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing `OC1M=111` in the `TIMx_CCMR1` register. Optionally the preload registers can be enabled by writing `OC1PE=1` in the `TIMx_CCMR1` register and `ARPE` in the `TIMx_CR1` register. In this case one has to write the compare value in the `TIMx_CCR1` register, the auto-reload value in the `TIMx_ARR` register, generate an update by setting the `UG` bit and wait for external trigger event on `TI2`. `CC1P` is written to '0 in this example.

In our example, the `DIR` and `CMS` bits in the `TIMx_CR1` register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the `OPM` bit in the `TIMx_CR1` register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When `OPM` bit in the `TIMx_CR1` register is set to '0', so the Repetitive Mode is selected.

#### Particular case: OCx fast enable:

In One-pulse mode, the edge detection on `TIx` input set the `CEN` bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{DELAY}$  min we can get.

If one wants to output a waveform with the minimum delay, the `OCxFE` bit can be set in the `TIMx_CCMRx` register. Then `OCxRef` (and `OCx`) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. `OCxFE` acts only if the channel is configured in PWM1 or PWM2 mode.

### 25.3.14 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in [Section 25.3.13](#):

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits `SMS[3:0] = '1000'` (Combined Reset + trigger mode) in the `TIMx_SMCR` register, and the `OCxM[3:0]` bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

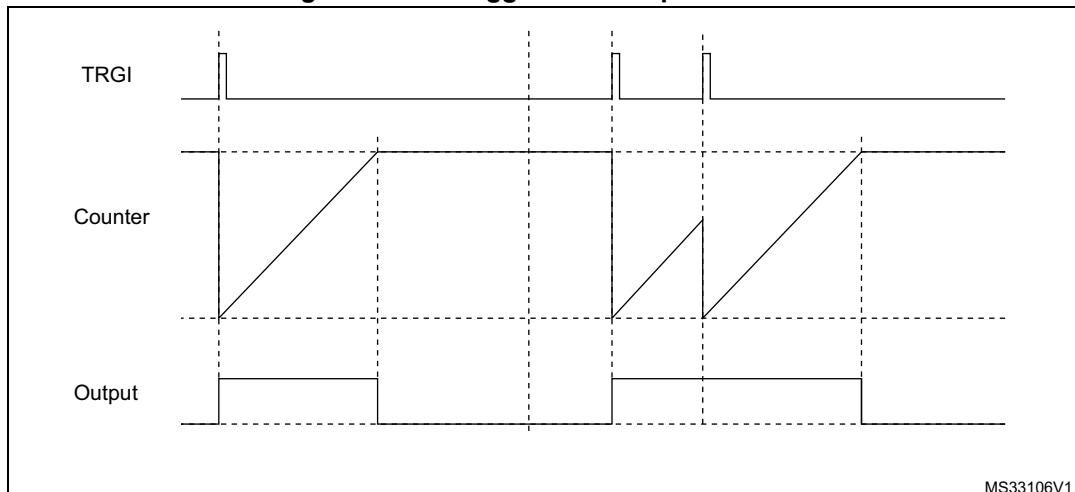
If the timer is configured in Up-counting mode, the corresponding `CCRx` must be set to 0 (the `ARR` register sets the pulse length). If the timer is configured in Down-counting mode `CCRx` must be above or equal to `ARR`.

*Note:* In retriggerable one pulse mode, the `CCxIF` flag is not significant.

*The `OCxM[3:0]` and `SMS[3:0]` bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.*

*This mode must not be used with center-aligned PWM modes. It is mandatory to have `CMS[1:0] = 00` in `TIMx_CR1`.*

Figure 240. Retriggerable one-pulse mode.



### 25.3.15 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx\_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx\_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to [Table 165](#). The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx\_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx\_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx\_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx\_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the-quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 165. Counting direction versus encoder signals

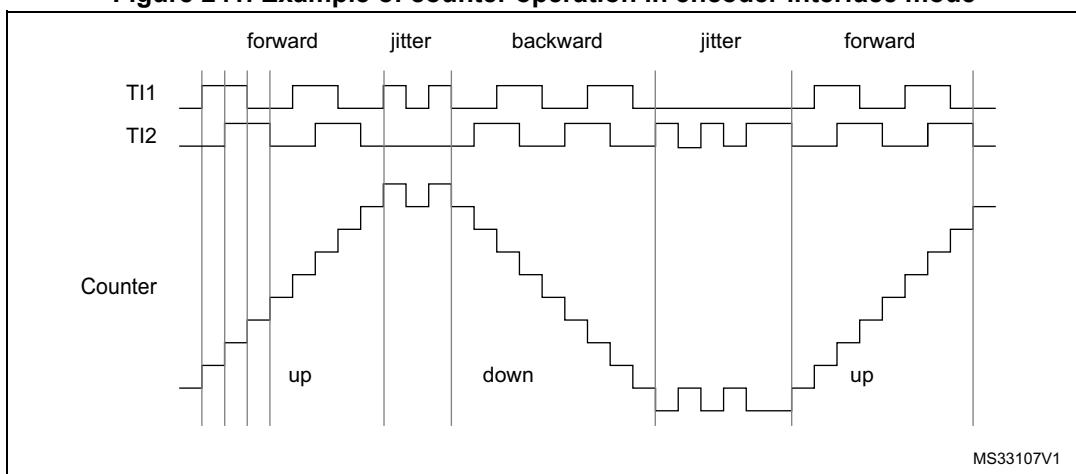
Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting 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 241* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

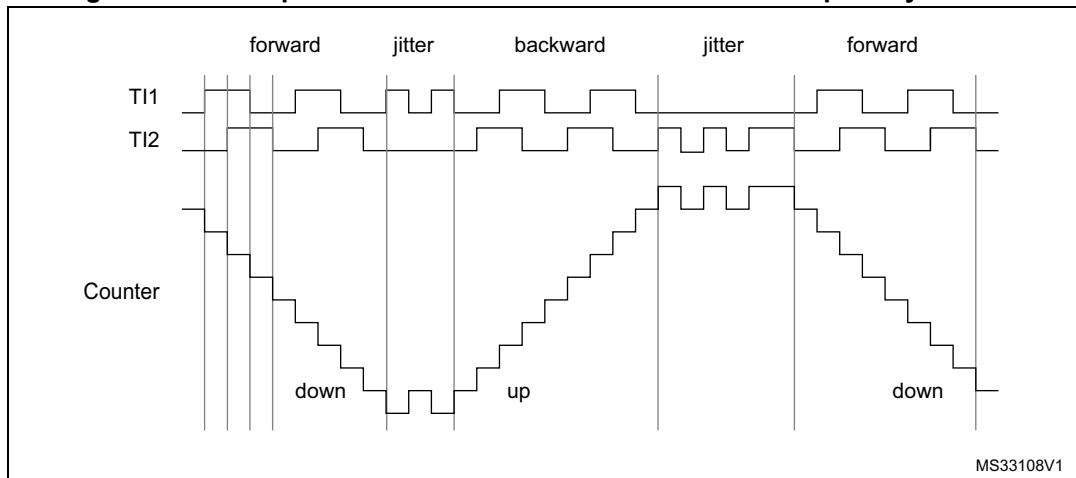
- CC1S= 01 (TIMx\_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx\_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = '0' (TIMx\_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = '0' (TIMx\_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx\_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx\_CR1 register, Counter is enabled)

Figure 241. Example of counter operation in encoder interface mode



*Figure 242* gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

**Figure 242. Example of encoder interface mode with TI1FP1 polarity inverted**



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

### 25.3.16 UIF bit remapping

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This permits to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

### 25.3.17 Timer input XOR function

The TI1S bit in the TIM1xx\_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx CH1 to TIMx CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in [Section 24.3.25: Interfacing with Hall sensors on page 713](#).

### 25.3.18 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

#### Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx\_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx\_ARR, TIMx\_CCRx) are updated.

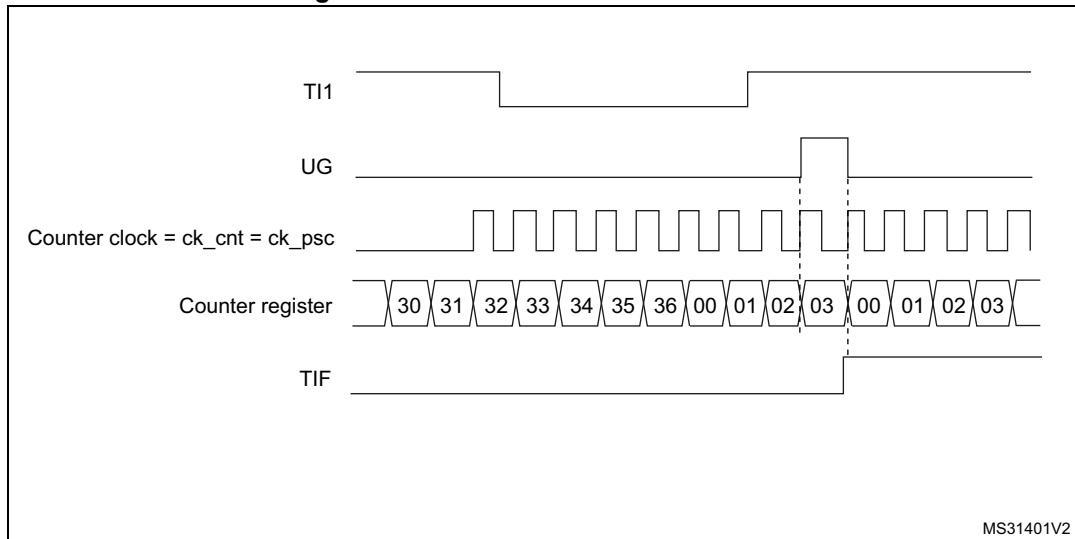
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx\_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx\_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx\_SR register) and an interrupt request 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 243. Control circuit in reset mode



#### Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

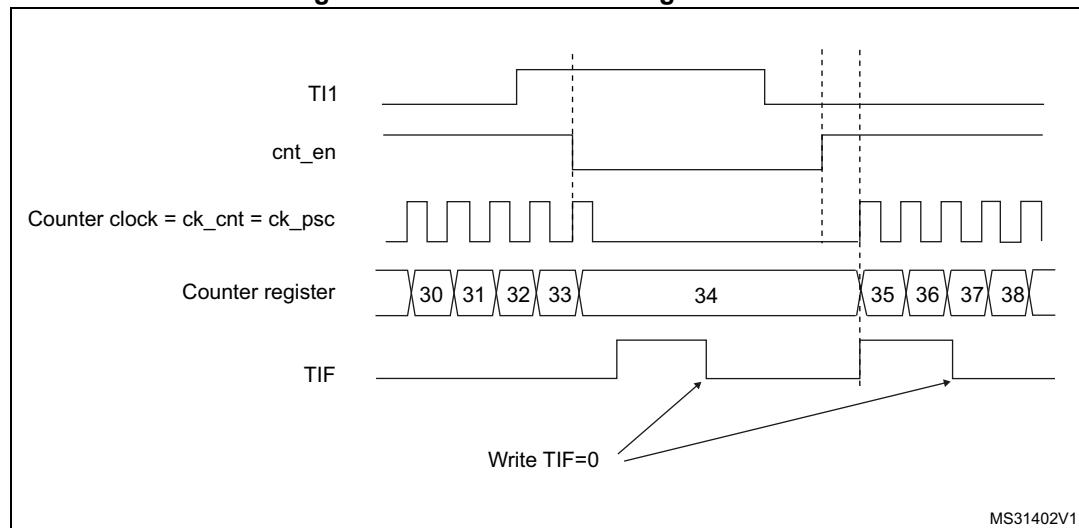
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx\_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx\_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx\_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 244. Control circuit in gated mode



1. The configuration “CCxP=CCxNP=1” (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Note:

*The configuration “CCxP=CCxNP=1” (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.*

### Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

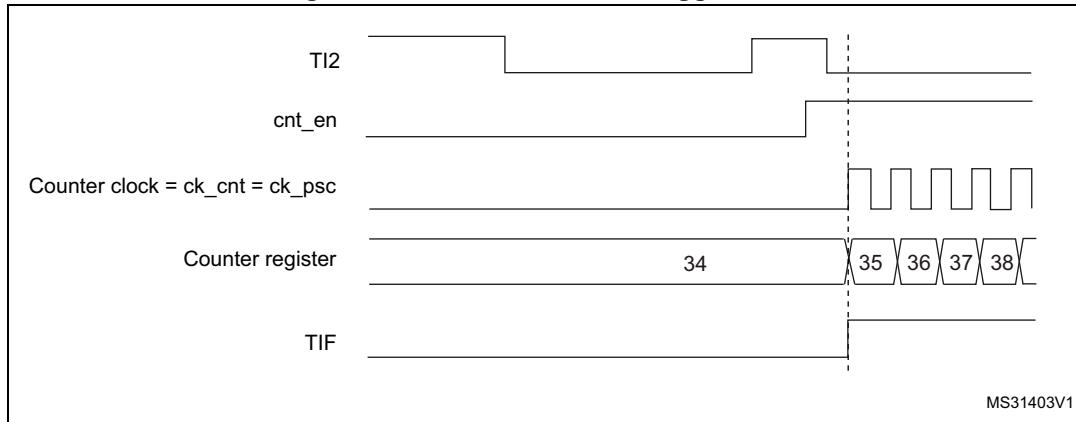
1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx\_CCMR1 register. Write

- CC2P=1 and CC2NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

**Figure 245. 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.

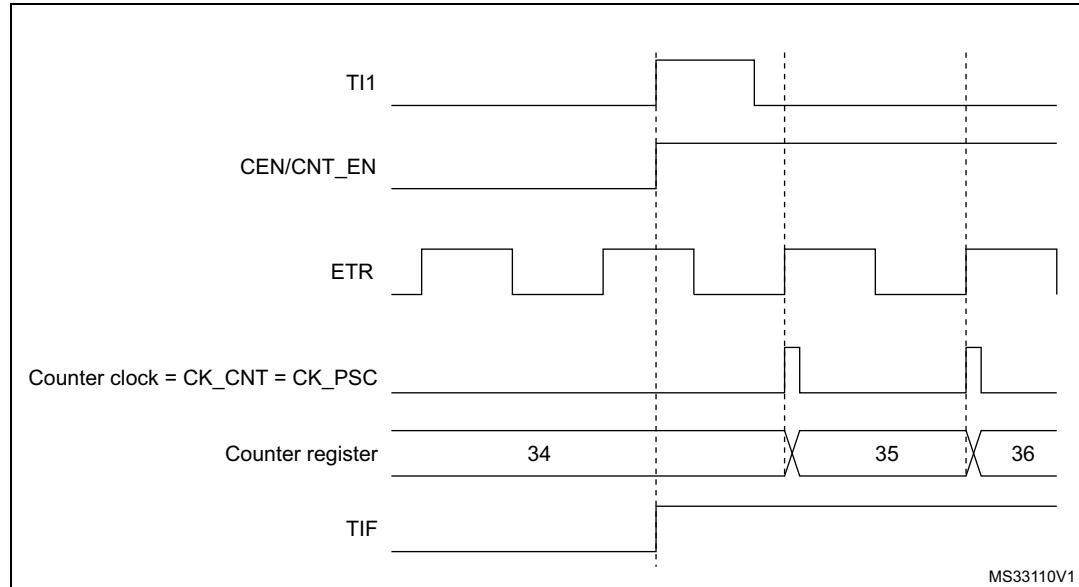
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

- 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.
- 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).
- Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

**Figure 246. Control circuit in external clock mode 2 + trigger mode**

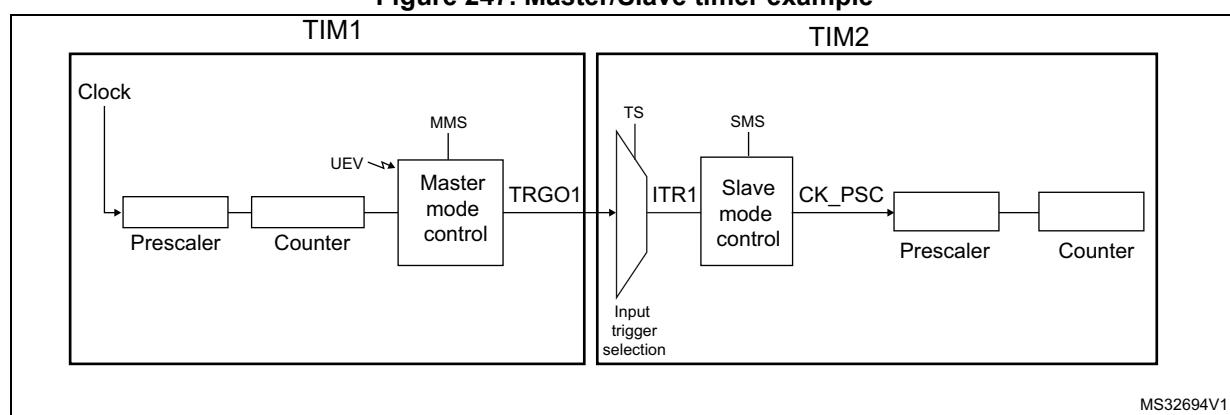


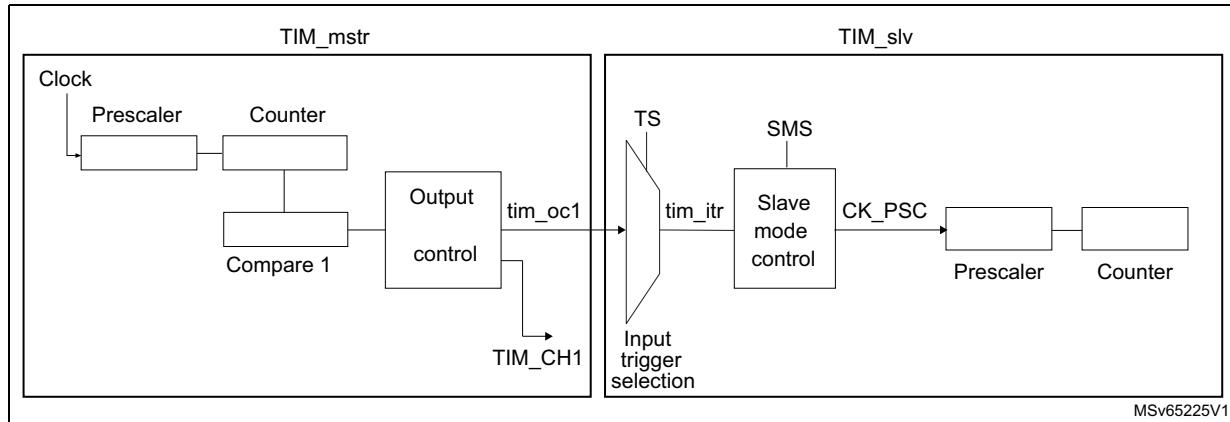
### 25.3.19 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

[Figure 247: Master/Slave timer example](#) and [Figure 248: Master/slave connection example with 1 channel only timers](#) present an overview of the trigger selection and the master mode selection blocks.

**Figure 247. Master/Slave timer example**



**Figure 248. Master/slave connection example with 1 channel only timers**

**Note:** The timers with one channel only (see [Figure 248](#)) do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx\_SMCR register on the device to identify which timers can be targeted as slave. The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer will detect the trigger. For instance, if the destination’s timer CK\_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

### Using one timer as prescaler for another timer

For example, TIM1 can be configured to act as a prescaler for TIM2. Refer to [Figure 247](#). To do this:

1. Configure TIM1 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM1\_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM1 to TIM2, TIM2 must be configured in slave mode using ITR0 as internal trigger. This is selected through the TS bits in the TIM2\_SMCR register (writing TS=00000).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM2\_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM1 trigger signal (which correspond to the TIM1 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx\_CR1 register).

**Note:** If OCx is selected on TIM1 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

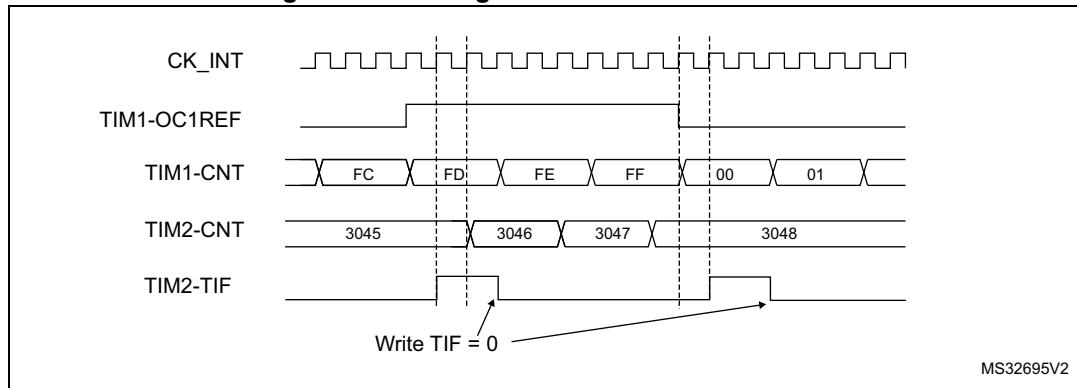
### Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 1. Refer to [Figure 247](#) for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM1 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK\_INT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

1. Configure TIM1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1\_CR2 register).
2. Configure the TIM1 OC1REF waveform (TIM1\_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM1 (TS=00000 in the TIM2\_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2\_SMCR register).
5. Enable TIM2 by writing '1 in the CEN bit (TIM\_CR1 register).
6. Start TIM by writing '1 in the CEN bit (TIM1\_CR1 register).

**Note:** *The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.*

**Figure 249. Gating TIM2 with OC1REF of TIM1**

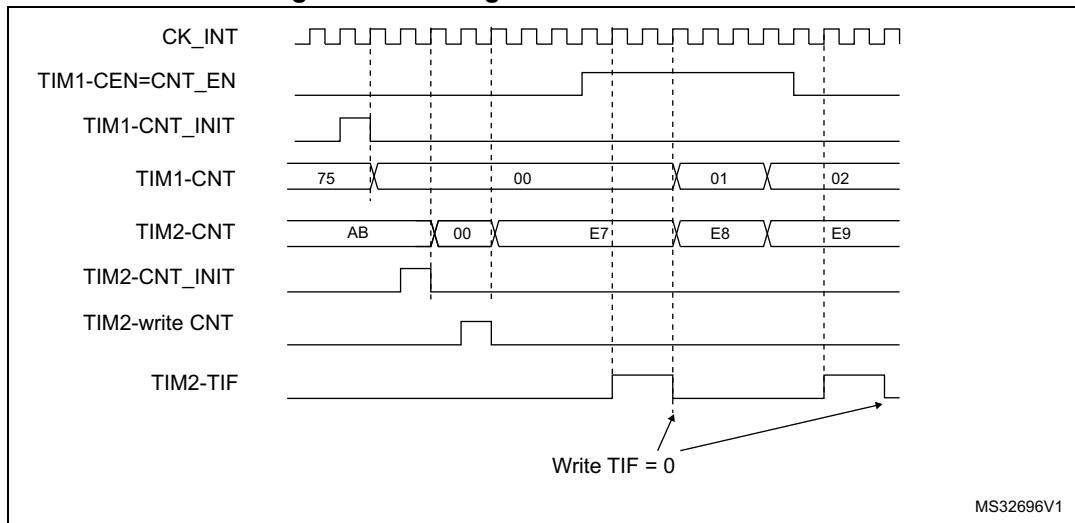


In the example in [Figure 249](#), the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM1. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx\_EGR registers.

In the next example (refer to [Figure 250](#)), we synchronize TIM1 and TIM2. TIM1 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM1 is disabled by writing '0 to the CEN bit in the TIM1\_CR1 register:

1. Configure TIM1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1\_CR2 register).
2. Configure the TIM1 OC1REF waveform (TIM1\_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM1 (TS=00000 in the TIM2\_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2\_SMCR register).
5. Reset TIM1 by writing '1 in UG bit (TIM1\_EGR register).
6. Reset TIM2 by writing '1 in UG bit (TIM2\_EGR register).
7. Initialize TIM2 to 0xE7 by writing '0xE7' in the TIM2 counter (TIM2\_CNTL).
8. Enable TIM2 by writing '1 in the CEN bit (TIM2\_CR1 register).
9. Start TIM1 by writing '1 in the CEN bit (TIM1\_CR1 register).
10. Stop TIM1 by writing '0 in the CEN bit (TIM1\_CR1 register).

Figure 250. Gating TIM2 with Enable of TIM1

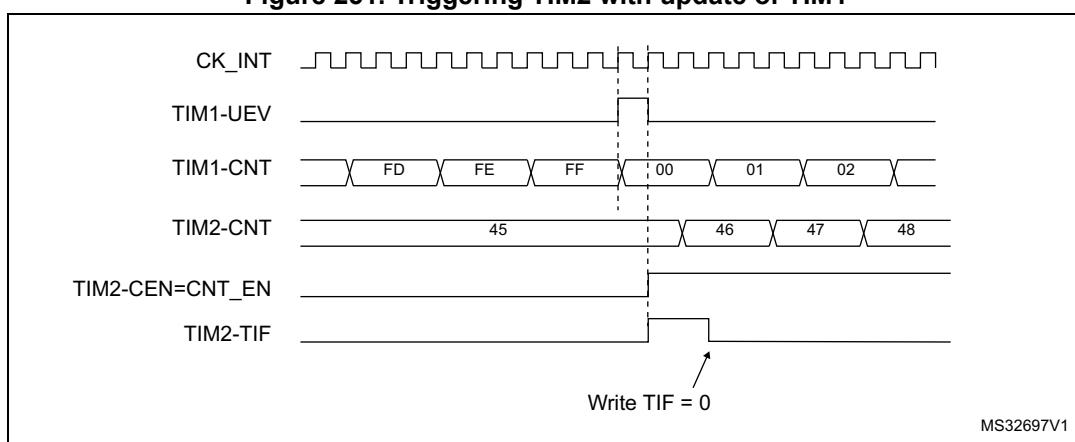


### Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 1. Refer to [Figure 247](#) for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write '0' to the CEN bit in the TIM2\_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK\_INT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

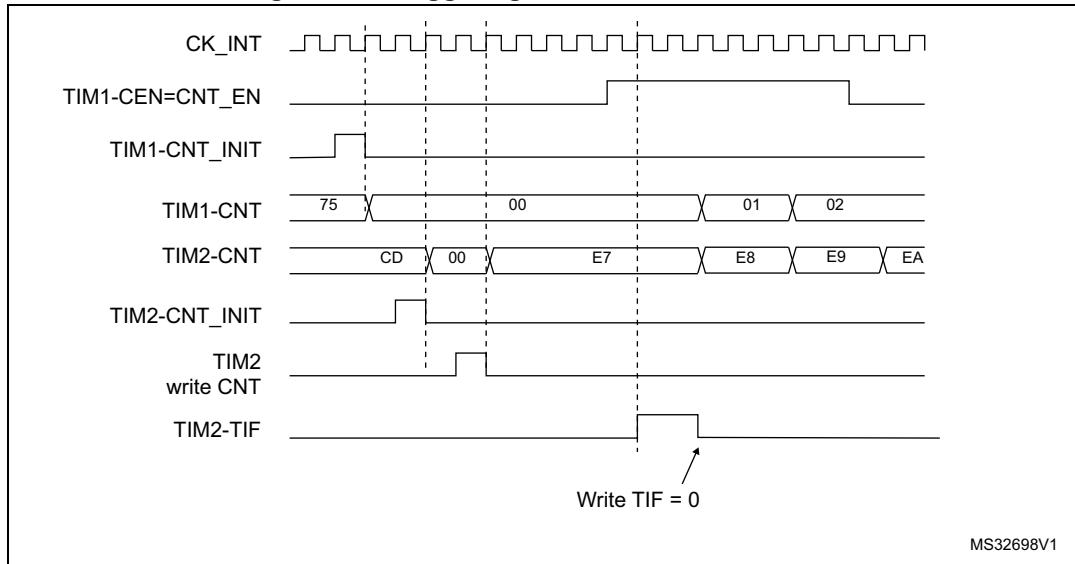
1. Configure TIM1 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM1\_CR2 register).
2. Configure the TIM1 period (TIM1\_ARR registers).
3. Configure TIM2 to get the input trigger from TIM1 (TS=00000 in the TIM2\_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2\_SMCR register).
5. Start TIM1 by writing '1 in the CEN bit (TIM1\_CR1 register).

Figure 251. Triggering TIM2 with update of TIM1



As in the previous example, both counters can be initialized before starting counting. [Figure 252](#) shows the behavior with the same configuration as in [Figure 251](#) but in trigger mode instead of gated mode (SMS=110 in the TIM2\_SMCR register).

**Figure 252. Triggering TIM2 with Enable of TIM1**



**Note:** The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

### 25.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx\_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx\_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx\_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx\_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register:

Example:

00000: TIMx\_CR1

00001: TIMx\_CR2

00010: TIMx\_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers ( $x = 2, 3, 4$ ) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
  - DMA channel peripheral address is the DMAR register address
  - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
  - Number of data to transfer = 3 (See note below).
  - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:  
DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

*Note:* A null value can be written to the reserved registers.

### 25.3.21 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on DBG\_TIM2\_STOP configuration bit in DBGMCU module. For more details, refer to [Section 41.8.3: DBGMCU CPU1 APB1 peripheral freeze register 1 \(DBGMCU\\_APB1FZR1\)](#).

## 25.4 TIM2 registers

In this section, “TIMx” should be understood as “TIM2” since there is only one instance of this type of timer for the products to which this reference manual applies.

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

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

### 25.4.1 TIM2 control register 1 (TIM2\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping

- 0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.
- 1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK\_INT) frequency and sampling clock used by the digital filters (ETR, TIx),

- 00:  $t_{DTS} = t_{CK\_INT}$
- 01:  $t_{DTS} = 2 \times t_{CK\_INT}$
- 10:  $t_{DTS} = 4 \times t_{CK\_INT}$
- 11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

- 0: TIMx\_ARR register is not buffered
- 1: TIMx\_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set both when the counter is counting up or down.

*Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)*

Bit 4 **DIR**: Direction

- 0: Counter used as upcounter
- 1: Counter used as downcounter

*Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.*

Bit 3 **OPM**: One-pulse mode

- 0: Counter is not stopped at update event
- 1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

- 0: Any of the following events generate an update interrupt or DMA request if enabled.
- These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

- 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

- 0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

- 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

- 0: Counter disabled
- 1: Counter enabled

*Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

CEN is cleared automatically in one-pulse mode, when an update event occurs.

## 25.4.2 TIM2 control register 2 (TIM2\_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.							

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

Bit 7 **TI1S**: TI1 selection

- 0: The TIMx\_CH1 pin is connected to TI1 input
  - 1: The TIMx\_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)
- See also [Section 24.3.25: Interfacing with Hall sensors on page 713](#)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits permit to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx\_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT\_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx\_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred.  
(TRGO)

100: **Compare** - OC1REFC signal is used as trigger output (TRGO)

101: **Compare** - OC2REFC signal is used as trigger output (TRGO)

110: **Compare** - OC3REFC signal is used as trigger output (TRGO)

111: **Compare** - OC4REFC signal is used as trigger output (TRGO)

*Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

Bit 3 **CCDS**: Capture/compare DMA selection

- 0: CCx DMA request sent when CCx event occurs
- 1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

### 25.4.3 TIM2 slave mode control register (TIM2\_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or  $\overline{ETR}$  is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge

1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).*

*It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).*

*If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of CK\_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at  $f_{DTS}$
- 0001:  $f_{SAMPLING} = f_{CK\_INT}$ , N=2
- 0010:  $f_{SAMPLING} = f_{CK\_INT}$ , N=4
- 0011:  $f_{SAMPLING} = f_{CK\_INT}$ , N=8
- 0100:  $f_{SAMPLING} = f_{DTS}/2$ , N=6
- 0101:  $f_{SAMPLING} = f_{DTS}/2$ , N=8
- 0110:  $f_{SAMPLING} = f_{DTS}/4$ , N=6
- 0111:  $f_{SAMPLING} = f_{DTS}/4$ , N=8
- 1000:  $f_{SAMPLING} = f_{DTS}/8$ , N=6
- 1001:  $f_{SAMPLING} = f_{DTS}/8$ , N=8
- 1010:  $f_{SAMPLING} = f_{DTS}/16$ , N=5
- 1011:  $f_{SAMPLING} = f_{DTS}/16$ , N=6
- 1100:  $f_{SAMPLING} = f_{DTS}/16$ , N=8
- 1101:  $f_{SAMPLING} = f_{DTS}/32$ , N=5
- 1110:  $f_{SAMPLING} = f_{DTS}/32$ , N=6
- 1111:  $f_{SAMPLING} = f_{DTS}/32$ , N=8

Bit 7 **MSM**: Master/Slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (ITR0)

00001: Internal Trigger 1 (ITR1)

00010: Internal Trigger 2 (ITR2)

00011: Internal Trigger 3 (ITR3)

00100: TI1 Edge Detector (TI1F\_ED)

00101: Filtered Timer Input 1 (TI1FP1)

00110: Filtered Timer Input 2 (TI2FP2)

00111: External Trigger input (ETRF)

01000: Internal Trigger 4 (ITR4)

01001: Internal Trigger 5 (ITR5)

01010: Internal Trigger 6 (ITR6)

01011: Internal Trigger 7 (ITR7)

01100: Internal Trigger 8 (ITR8)

Others: Reserved

See [Table 166: TIM2 internal trigger connection on page 812](#) for more details on ITRx meaning for each Timer.

*Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.*

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source

0: OCREF\_CLR\_INT is connected to the OCREF\_CLR input

1: OCREF\_CLR\_INT is connected to ETRF

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

0000: Slave mode disabled - if CEN = '1 then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

*Note: The gated mode must not be used if TI1F\_ED is selected as the trigger input (TS=00100). Indeed, TI1F\_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.*

*Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.*

**Table 166. TIM2 internal trigger connection**

Slave TIM	ITR0	ITR1	ITR2 - ITR8
TIM2	TIM1	USB_FS_SOF <sup>(1)</sup>	-

1. This connection is valid only when the ITR\_RMP bit is set in the TIM2\_OR1 register.

#### 25.4.4 TIM2 DMA/Interrupt enable register (TIM2\_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable

0: CC4 DMA request disabled.

1: CC4 DMA request enabled.

- Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable  
0: CC3 DMA request disabled.  
1: CC3 DMA request enabled.
- Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable  
0: CC2 DMA request disabled.  
1: CC2 DMA request enabled.
- Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable  
0: CC1 DMA request disabled.  
1: CC1 DMA request enabled.
- Bit 8 **UDE**: Update DMA request enable  
0: Update DMA request disabled.  
1: Update DMA request enabled.
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **TIE**: Trigger interrupt enable  
0: Trigger interrupt disabled.  
1: Trigger interrupt enabled.
- Bit 5 Reserved, must be kept at reset value.
- Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable  
0: CC4 interrupt disabled.  
1: CC4 interrupt enabled.
- Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable  
0: CC3 interrupt disabled.  
1: CC3 interrupt enabled.
- Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable  
0: CC2 interrupt disabled.  
1: CC2 interrupt enabled.
- Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable  
0: CC1 interrupt disabled.  
1: CC1 interrupt enabled.
- Bit 0 **UIE**: Update interrupt enable  
0: Update interrupt disabled.  
1: Update interrupt enabled.

### 25.4.5 TIM2 status register (TIM2\_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag  
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag  
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag  
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag  
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.  
0: No overcapture has been detected.  
1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

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

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.  
0: No trigger event occurred.  
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag  
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag  
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx\_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

**If channel CC1 is configured as output:** this flag is set when the content of the counter TIMx\_CNT matches the content of the TIMx\_CCR1 register. When the content of TIMx\_CCR1 is greater than the content of TIMx\_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx\_CR1 register for the full description.

**If channel CC1 is configured as input:** this bit is set when counter value has been captured in TIMx\_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx\_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred

1: Update interrupt pending. This bit is set by hardware when the registers are updated: At overflow or underflow and if UDIS=0 in the TIMx\_CR1 register.

When CNT is reinitialized by software using the UG bit in TIMx\_EGR register, if URS=0 and UDIS=0 in the TIMx\_CR1 register.

When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx\_CR1 register.

## 25.4.6 TIM2 event generation register (TIM2\_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG								

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

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx\_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

**If channel CC1 is configured as output:**

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

**If channel CC1 is configured as input:**

The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx\_ARR) if DIR=1 (downcounting).

### 25.4.7 TIM2 capture/compare mode register 1 [alternate] (TIM2\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

#### Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx\_CCER).*

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at  $f_{DTS}$

0001:  $f_{SAMPLING} = f_{CK\_INT}$ , N=2

0010:  $f_{SAMPLING} = f_{CK\_INT}$ , N=4

0011:  $f_{SAMPLING} = f_{CK\_INT}$ , N=8

0100:  $f_{SAMPLING} = f_{DTS}/2$ , N=6

0101:  $f_{SAMPLING} = f_{DTS}/2$ , N=8

0110:  $f_{SAMPLING} = f_{DTS}/4$ , N=6

0111:  $f_{SAMPLING} = f_{DTS}/4$ , N=8

1000:  $f_{SAMPLING} = f_{DTS}/8$ , N=6

1001:  $f_{SAMPLING} = f_{DTS}/8$ , N=8

1010:  $f_{SAMPLING} = f_{DTS}/16$ , N=5

1011:  $f_{SAMPLING} = f_{DTS}/16$ , N=6

1100:  $f_{SAMPLING} = f_{DTS}/16$ , N=8

1101:  $f_{SAMPLING} = f_{DTS}/32$ , N=5

1110:  $f_{SAMPLING} = f_{DTS}/32$ , N=6

1111:  $f_{SAMPLING} = f_{DTS}/32$ , N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (TIMx\_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx\_CCER).*

### 25.4.8 TIM2 capture/compare mode register 1 [alternate] (TIM2\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

#### Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx\_CCER).*

Bit 7 **OC1CE**: Output compare 1 clear enable

0: OC1Ref is not affected by the ETRF input

1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx\_CNT<TIMx\_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0) as long as TIMx\_CNT>TIMx\_CCR1 else active (OC1REF=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx\_CNT<TIMx\_CCR1 else active. In downcounting, channel 1 is active as long as TIMx\_CNT>TIMx\_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

*Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.*

*Note: The OC1M[3] bit is not contiguous, located in bit 16.*

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx\_CCR1 enabled. Read/Write operations access the preload register. TIMx\_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx\_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx\_CCER).*

## 25.4.9 TIM2 capture/compare mode register 2 [alternate] (TIM2\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

### Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx\_CCER).*

Bits 7:4 **IC3F[3:0]**: Input capture 3 filterBits 3:2 **IC3PSC[1:0]**: Input capture 3 prescalerBits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx\_CCER).*

**25.4.10 TIM2 capture/compare mode register 2 [alternate] (TIM2\_CCMR2)**

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Output compare mode:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]	OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode

Refer to OC1M description (bits 6:4 in TIMx\_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx\_CCER).*

Bit 7 **OC3CE**: Output compare 3 clear enableBits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

Refer to OC1M description (bits 6:4 in TIMx\_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enableBit 2 **OC3FE**: Output compare 3 fast enableBits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx\_CCER).*

### 25.4.11 TIM2 capture/compare enable register (TIM2\_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E

Bit 15 **CC4NP**: Capture/Compare 4 output Polarity.

Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output Polarity.

Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable.

Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 output Polarity.

Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: Capture/Compare 3 output Polarity.

Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable.

Refer to CC1E description

- Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*  
Refer to CC1NP description
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*  
refer to CC1P description
- Bit 4 **CC2E**: *Capture/Compare 2 output enable.*  
Refer to CC1E description
- Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*  
**CC1 channel configured as output:** CC1NP must be kept cleared in this case.  
**CC1 channel configured as input:** This bit is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity. refer to CC1P description.
- Bit 2 Reserved, must be kept at reset value.
- Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*  
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)  
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)  
**When CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.  
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).  
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).  
CC1NP=1, CC1P=1: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.  
CC1NP=1, CC1P=0: This configuration is reserved, it must not be used.
- Bit 0 **CC1E**: *Capture/Compare 1 output enable.*  
0: Capture mode disabled / OC1 is not active  
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**Table 167. Output control bit for standard OCx channels**

CCxE bit	OCx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

**Note:** The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO control and alternate function registers.

#### 25.4.12 TIM2 counter [alternate] (TIM2\_CNT)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx\_CR1 register:

- This section is for UIFREMAP = 0
- Next section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

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

Bits 31:0 **CNT[31:0]**: counter value

#### 25.4.13 TIM2 counter [alternate] (TIM2\_CNT)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx\_CR1 register:

- Previous section is for UIFREMAP = 0
- This section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx\_ISR register

Bits 30:0 **CNT[30:0]**: counter value

#### 25.4.14 TIM2 prescaler (TIM2\_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 valueThe counter clock frequency CK\_CNT is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

### 25.4.15 TIM2 auto-reload register (TIM2\_ARR)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

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

Bits 31:0 **ARR[31:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 25.3.1: Time-base unit on page 764](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 25.4.16 TIM2 capture/compare register 1 (TIM2\_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/Compare 1 value

**If channel CC1 is configured as output:**

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC1 output.

**If channel CC1 is configured as input:**

CCR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx\_CCR1 register is read-only and cannot be programmed.

### 25.4.17 TIM2 capture/compare register 2 (TIM2\_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/Compare 2 value

**If channel CC2 is configured as output:**

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC2 output.

**If channel CC2 is configured as input:**

CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx\_CCR2 register is read-only and cannot be programmed.

#### 25.4.18 TIM2 capture/compare register 3 (TIM2\_CCR3)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
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 31:0 **CCR3[31:0]**: Capture/Compare value

**If channel CC3 is configured as output:**

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC3 output.

**If channel CC3 is configured as input:**

CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx\_CCR3 register is read-only and cannot be programmed.

#### 25.4.19 TIM2 capture/compare register 4 (TIM2\_CCR4)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
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 31:0 **CCR4[31:0]**: Capture/Compare value

1. if CC4 channel is configured as output (CC4S bits):  
CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.  
The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC4 output.
2. if CC4 channel is configured as input (CC4S bits in TIMx\_CCMR4 register):  
CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx\_CCR4 register is read-only and cannot be programmed.

#### 25.4.20 TIM2 DMA control register (TIM2\_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address).

00000: 1 transfer,  
00001: 2 transfers,  
00010: 3 transfers,  
...  
10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

Example:  
00000: TIMx\_CR1  
00001: TIMx\_CR2  
00010: TIMx\_SMCR  
...

**Example:** Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx\_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx\_CR1 address.

### 25.4.21 TIM2 DMA address for full transfer (TIM2\_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx\_CR1 address) + (DBA + DMA index) x 4

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

### 25.4.22 TIM2 option register 1 (TIM2\_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TI4_RMP	ETR_RMP	ITR_RMP	rw											

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

Bits 3:2 **TI4\_RMP**: Timer input 4 remap

Set and cleared by software.

00: TIM2 TI4 is connected to GPIO: Refer to Alternate Function mapping

01: TIM2 TI4 is connected to COMP1\_OUT

10: TIM2 TI4 is connected to COMP2\_OUT

11: TIM2 TI4 is connected to a logical OR between COMP1\_OUT and COMP2\_OUT

Bit 1 **ETR\_RMP**: External trigger 1 remap

Set and cleared by software.

0: TIM2 ETR is connected to GPIO: Refer to Alternate Function mapping

1: LSE internal clock is connected to TIM2\_ETR input

Bit 0 **ITR\_RMP**: Internal trigger remap

0: TIM2 Internal trigger ITR2 is not connected

1: TIM2 Internal trigger ITR2 is connected to USB\_FS\_SOF

### 25.4.23 TIM2 alternate function option register 1 (TIM2\_AF1)

Address offset: 0x60

Reset value: 0x0000 0000

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

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.

0000: GPIO or LSE internal clock, as per ETR\_RMP bit in TIM2\_OR1

0001: COMP1

0010: COMP2

Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

#### 25.4.24 TIM2 timer input selection register (TIM2\_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

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

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection

These bits select the TI2[0] to TI2[15] input source.

0000: TIM2\_CH2 input

Others: Reserved

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

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection

These bits select the TI1[0] to TI1[15] input source.

0000: TIM2\_CH1 input

Others: Reserved

## 25.4.25 TIMx register map

TIMx registers are mapped as described in the table below:

**Table 168. TIM2 register map and reset values**

Table 168. TIM2 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	<b>TIMx_CNT</b>	CNT[31] or UIFCPY	CNT[30:0]																														
			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x28	<b>TIMx_PSC</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
			0	0	0	0	0	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	<b>TIMx_ARR</b>	ARR[31:0]																															
		Reset value																															
0x30		Reserved																															
0x34	<b>TIMx_CCR1</b>	CCR1[31:0]																															
		Reset value																															
0x38	<b>TIMx_CCR2</b>	CCR2[31:0]																															
		Reset value																															
0x3C	<b>TIMx_CCR3</b>	CCR3[31:0]																															
		Reset value																															
0x40	<b>TIMx_CCR4</b>	CCR4[31:0]																															
		Reset value																															
0x44		Reserved																															
0x48	<b>TIMx_DCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
		Reset value	0	0	0	0	0	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	<b>TIMx_DMAR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
		Reset value	0	0	0	0	0	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	<b>TIM2_OR1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
		Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x60	<b>TIM2_AF1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
		Reset value	0	0	0	0	0	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 168. TIM2 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x68	TIM2_TISEL	Res.																															
	Reset value																					0	0	0	0	0	0	0	0	0	0		

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 26 General-purpose timers (TIM16/TIM17)

### 26.1 TIM16/TIM17 introduction

The TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

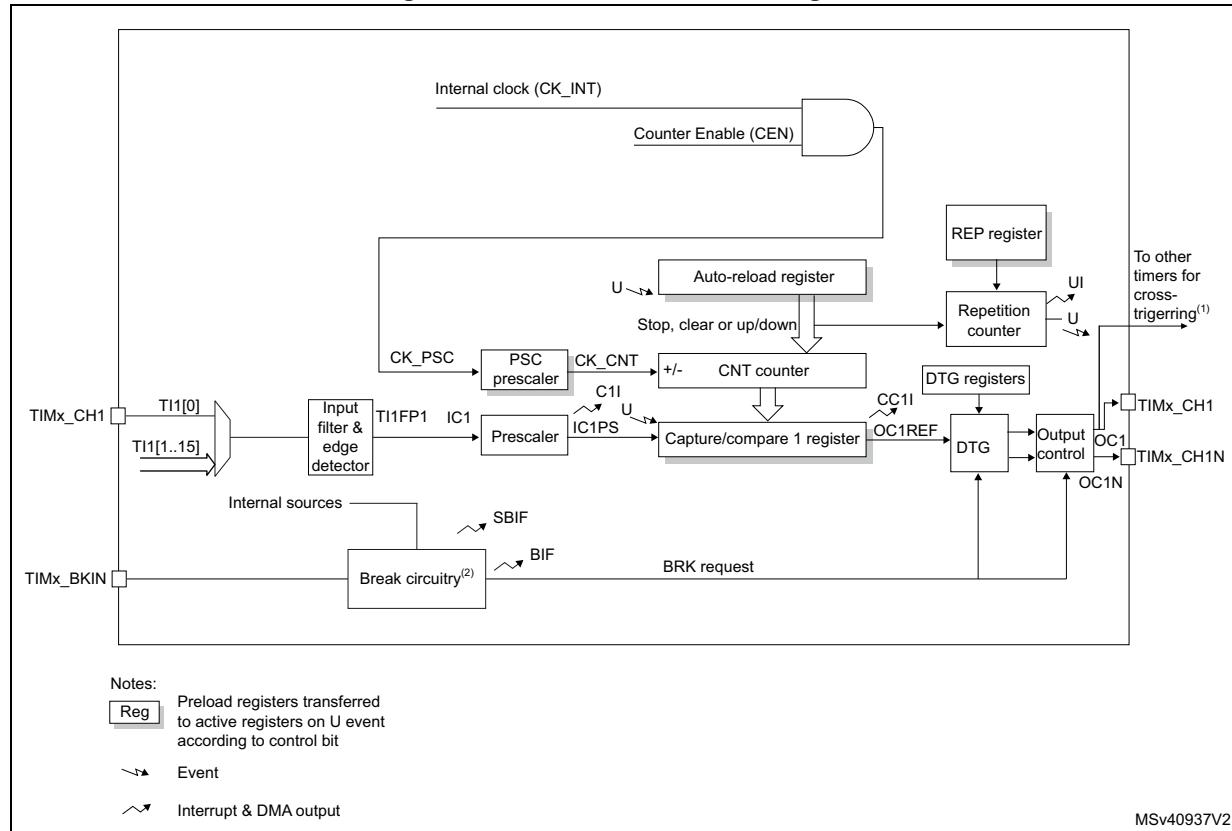
The TIM16/TIM17 timers are completely independent, and do not share any resources.

### 26.2 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

Figure 253. TIM16/TIM17 block diagram



1. This signal can be used as trigger for some slave timer, see [Section 26.3.18: Using timer output as trigger for other timers \(TIM16/TIM17\)](#).

## 26.3 TIM16/TIM17 functional description

### 26.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Auto-reload register (TIMx\_ARR)
- Repetition counter register (TIMx\_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx\_CR1 register.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 254* and *Figure 255* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 254. Counter timing diagram with prescaler division change from 1 to 2

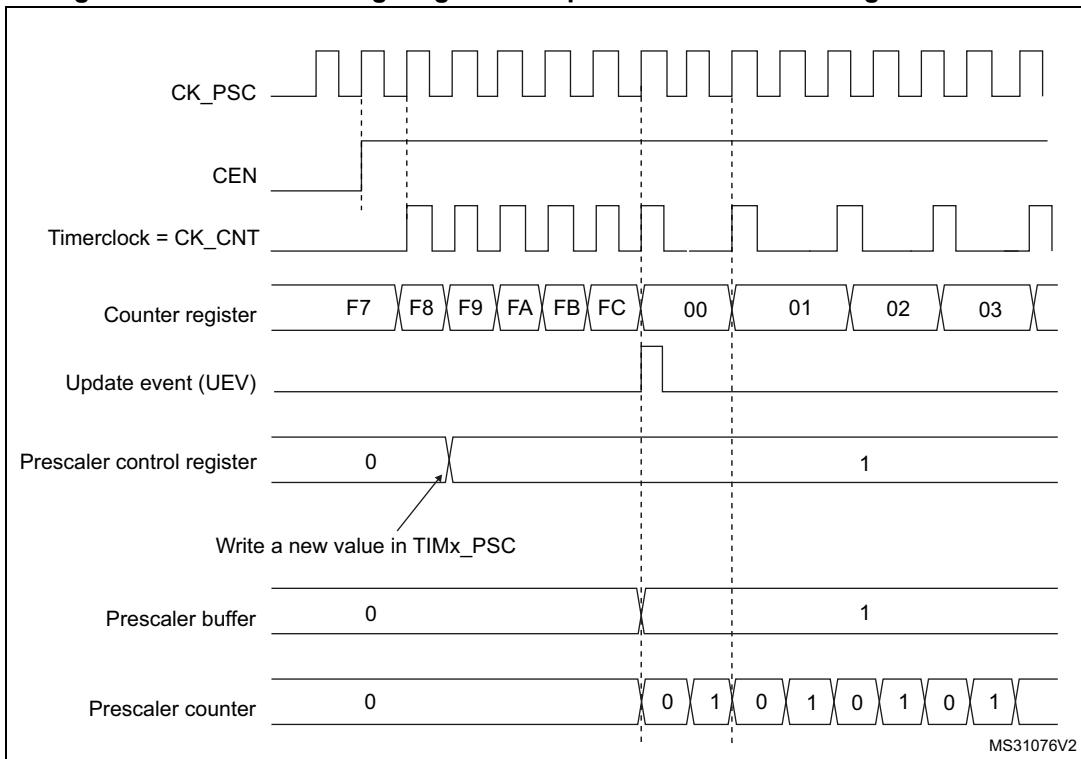
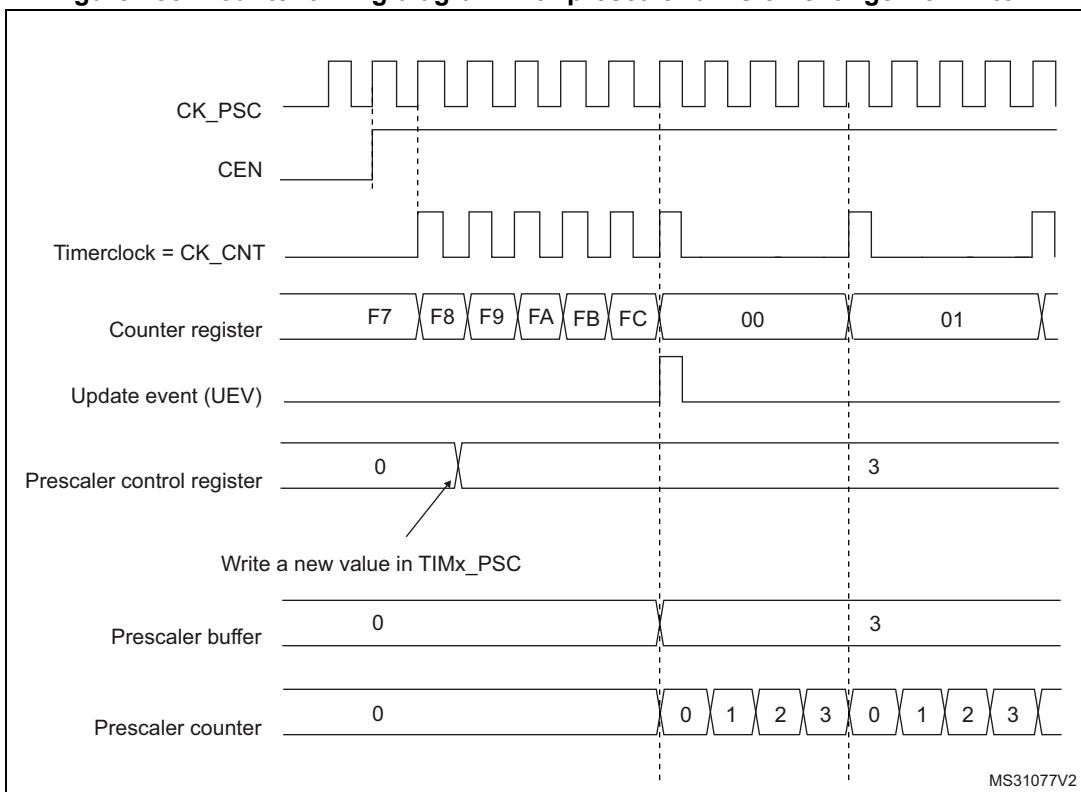


Figure 255. Counter timing diagram with prescaler division change from 1 to 4



### 26.3.2 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 256. Counter timing diagram, internal clock divided by 1

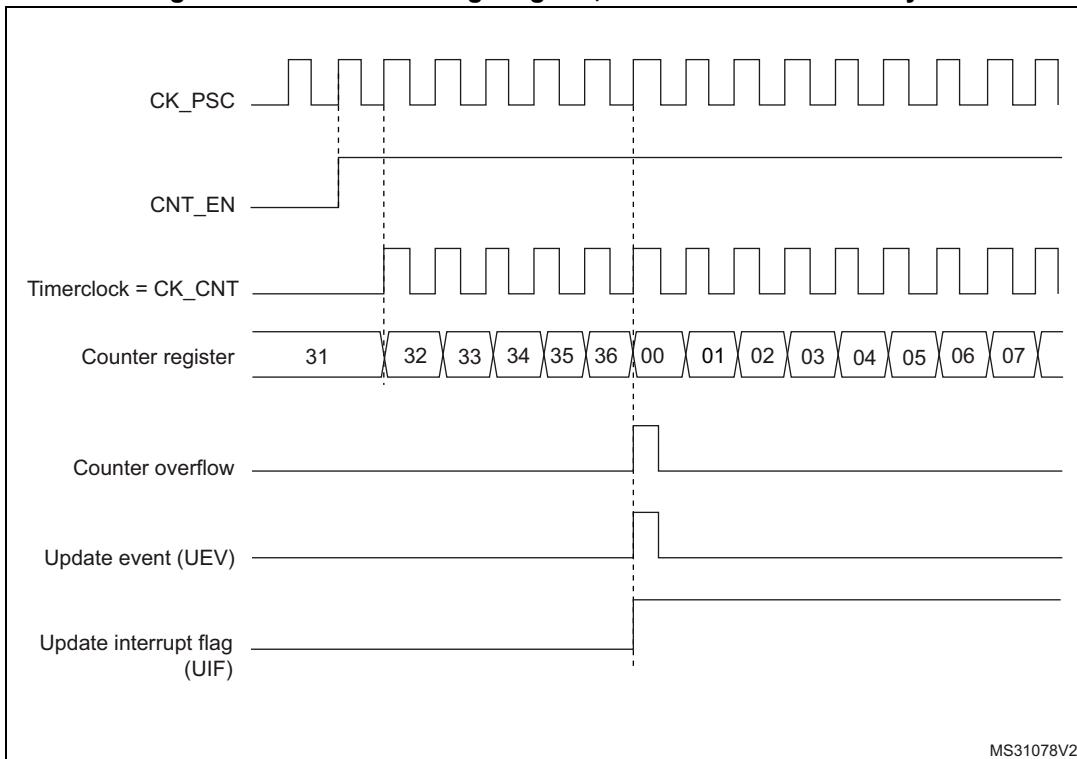


Figure 257. Counter timing diagram, internal clock divided by 2

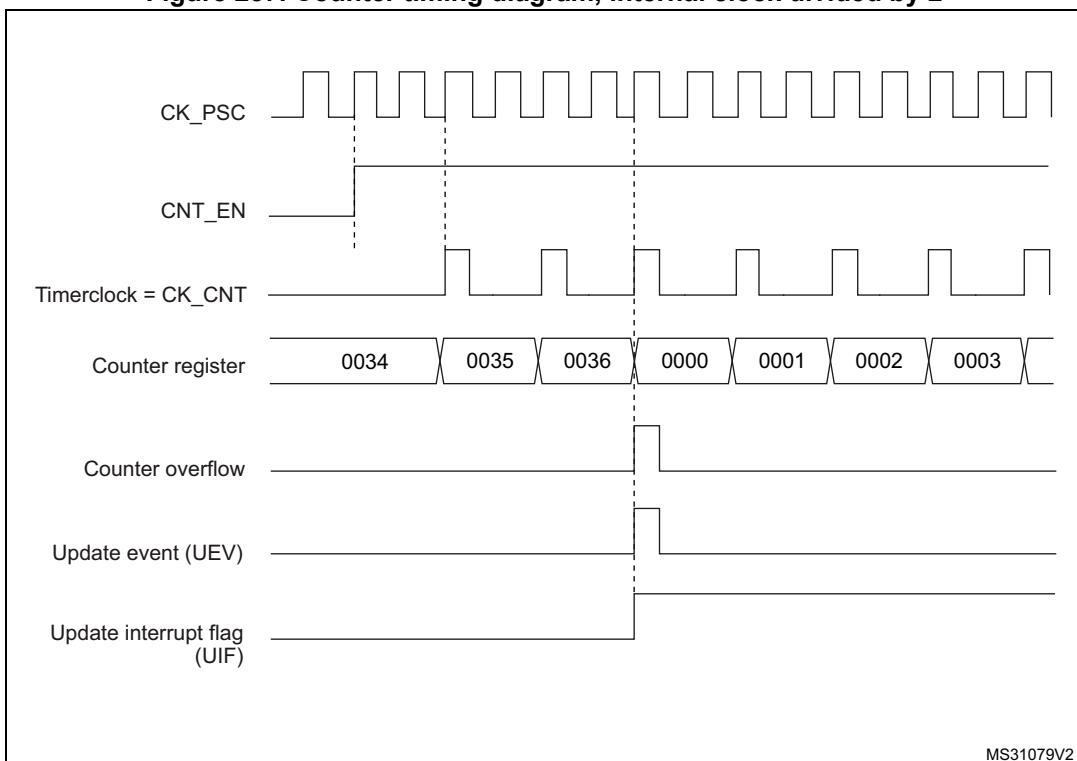


Figure 258. Counter timing diagram, internal clock divided by 4

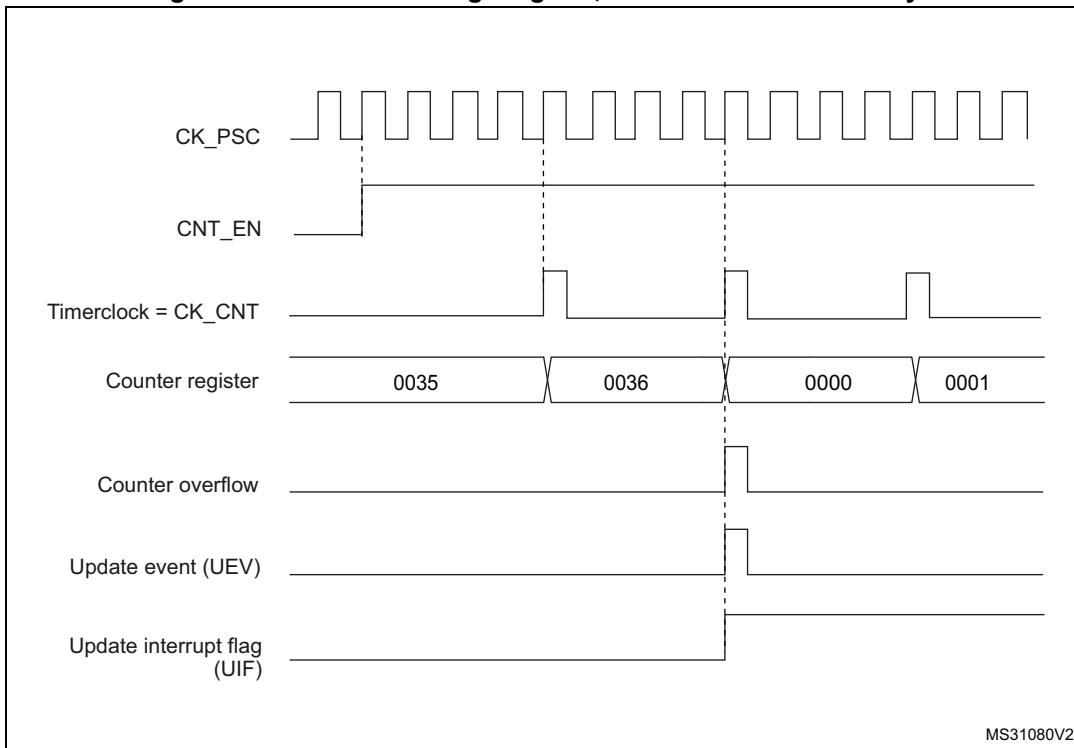
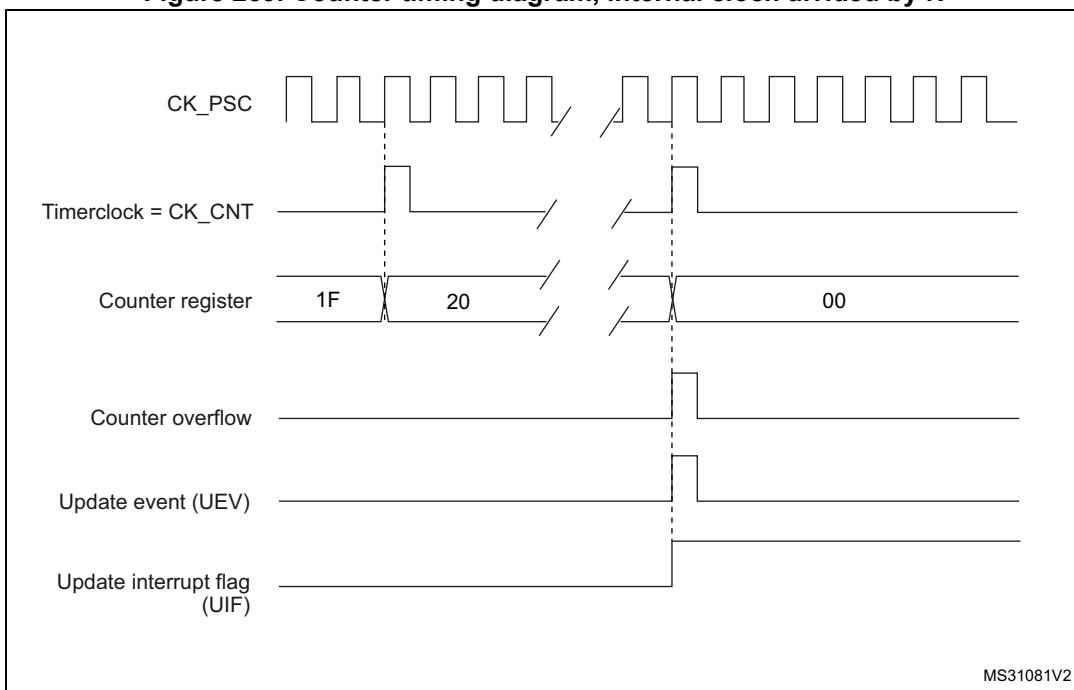
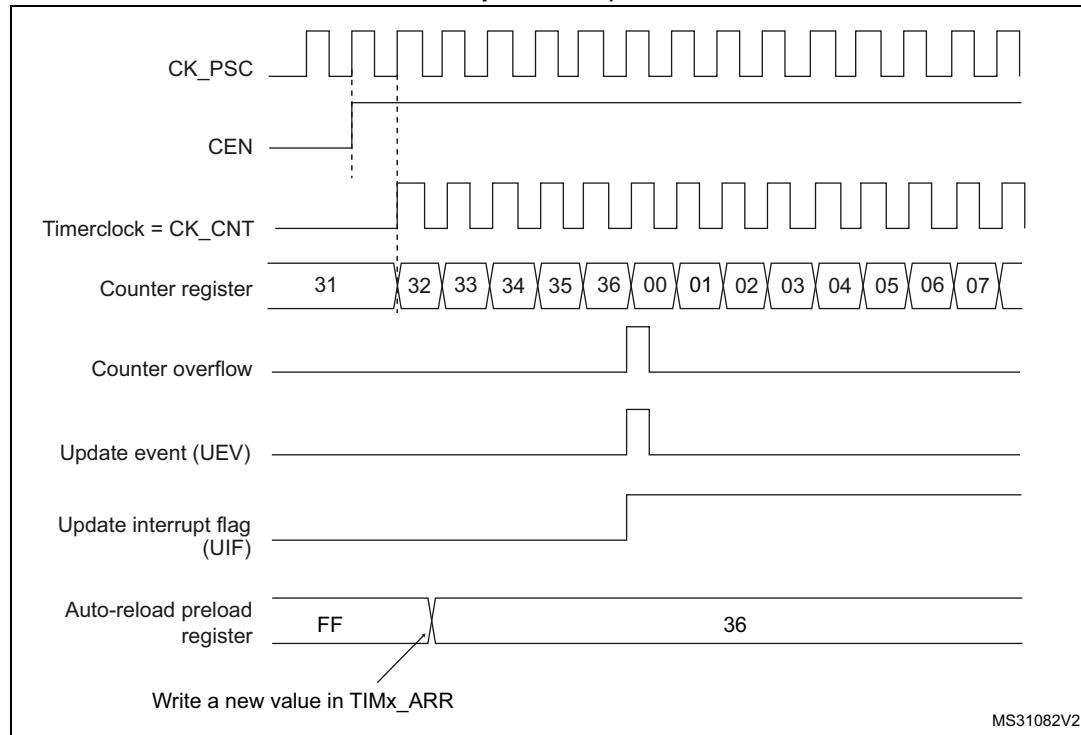


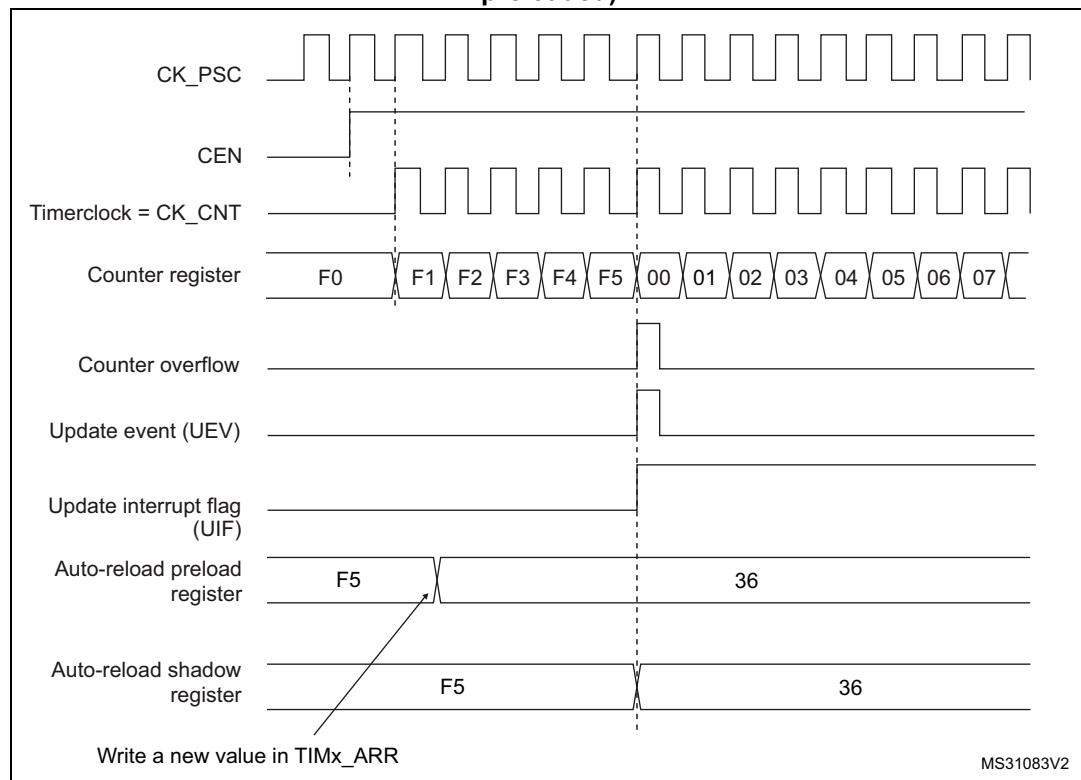
Figure 259. Counter timing diagram, internal clock divided by N



**Figure 260. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)**



**Figure 261. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



### 26.3.3 Repetition counter

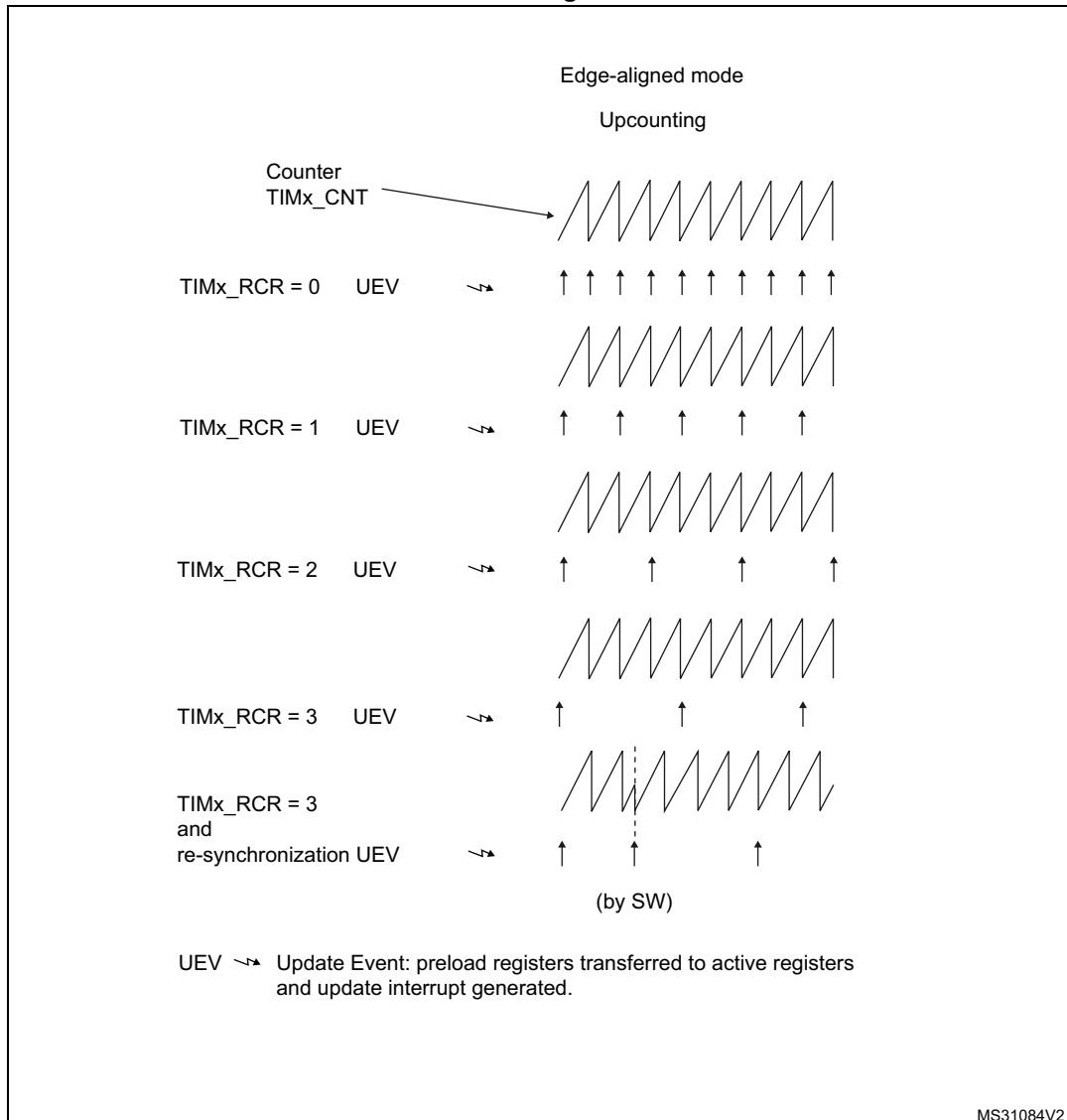
*Section 26.3.1: Time-base unit* describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx\_ARR auto-reload register, TIMx\_PSC prescaler register, but also TIMx\_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx\_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx\_RCR register value (refer to [Figure 262](#)). When the update event is generated by software (by setting the UG bit in TIMx\_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx\_RCR register.

**Figure 262. Update rate examples depending on mode and TIMx\_RCR register settings**



### 26.3.4 Clock selection

The counter clock can be provided by the following clock sources:

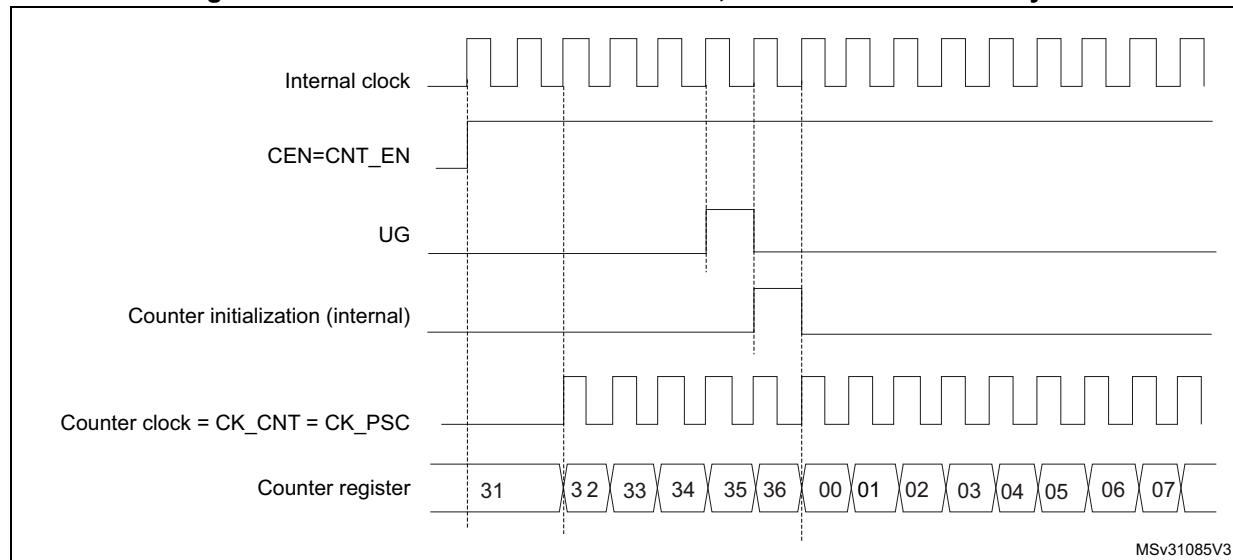
- Internal clock (CK\_INT)
  - External clock mode1: external input pin

### Internal clock source (CK\_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK\_INT.

*Figure 263* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

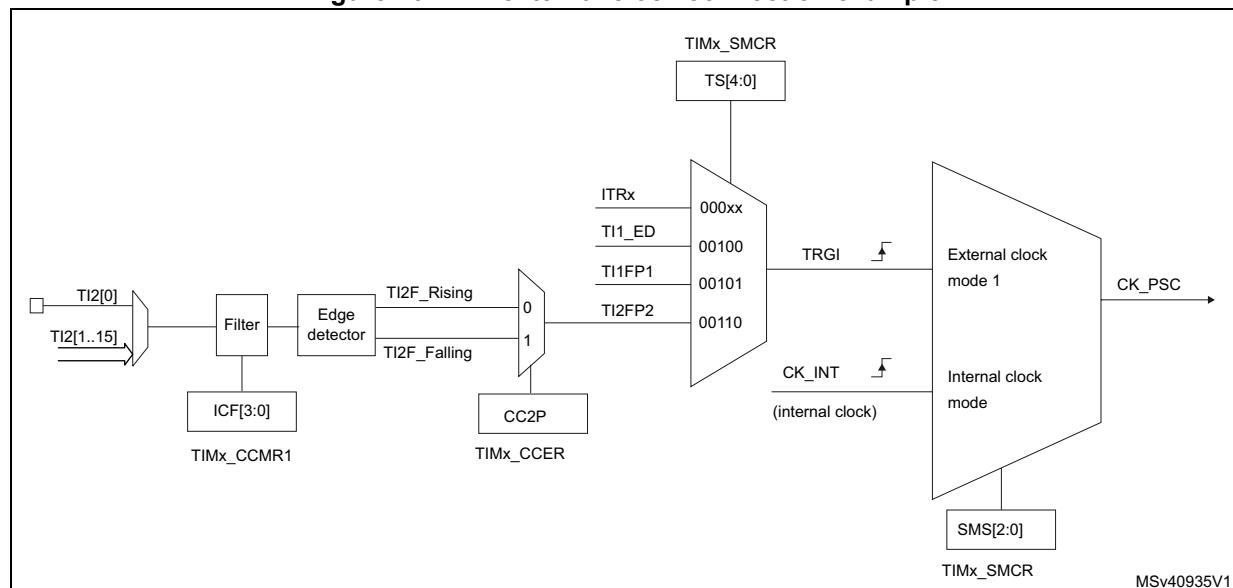
**Figure 263. Control circuit in normal mode, internal clock divided by 1**



### 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 264. TI2 external clock connection example**



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx\_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 in the TIMx\_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx\_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx\_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

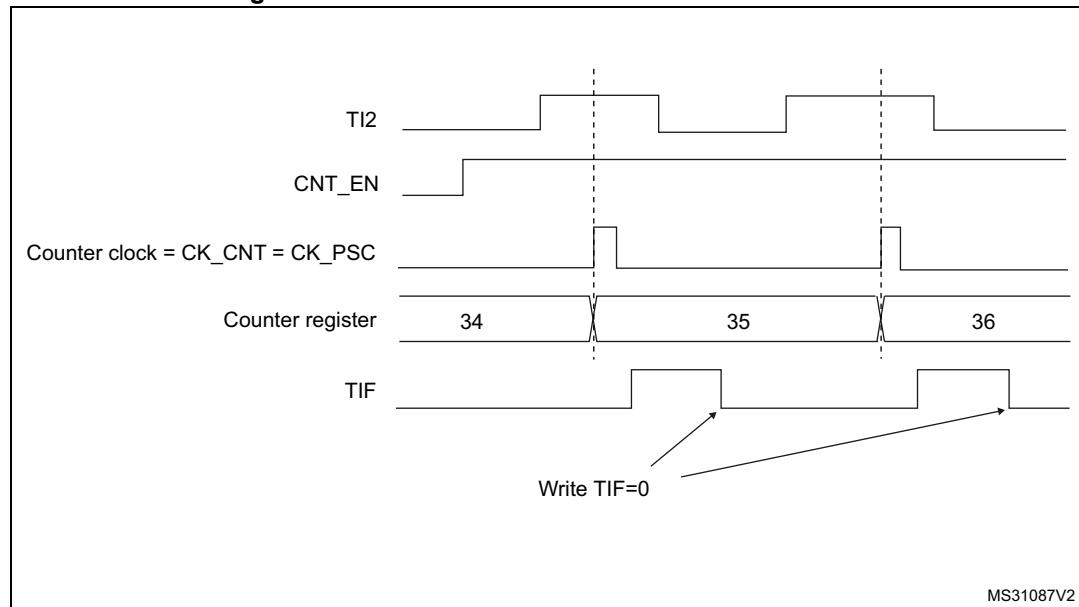
*Note:*

*The capture prescaler is not used for triggering, so it does not need to be configured.*

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

**Figure 265. Control circuit in external clock mode 1**



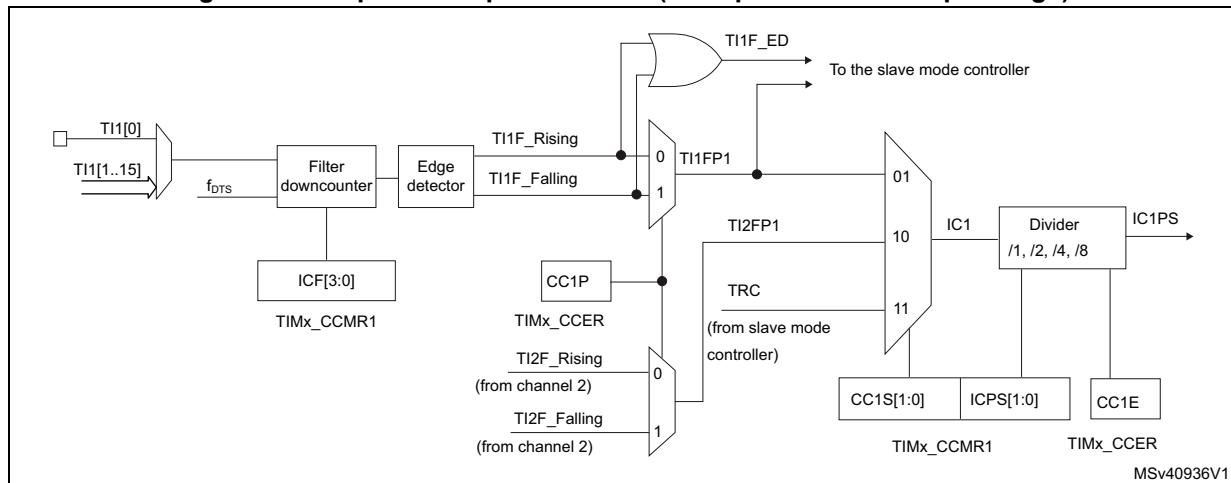
### 26.3.5 Capture/compare channels

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 266 to Figure 268* 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 266. 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 267. Capture/compare channel 1 main circuit

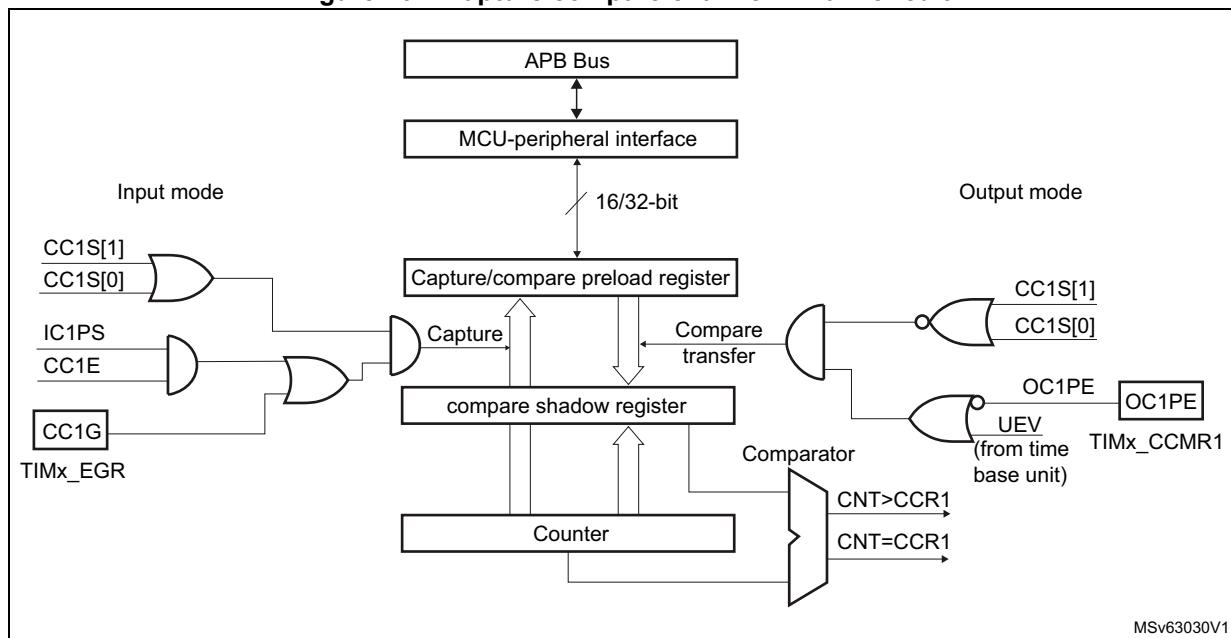
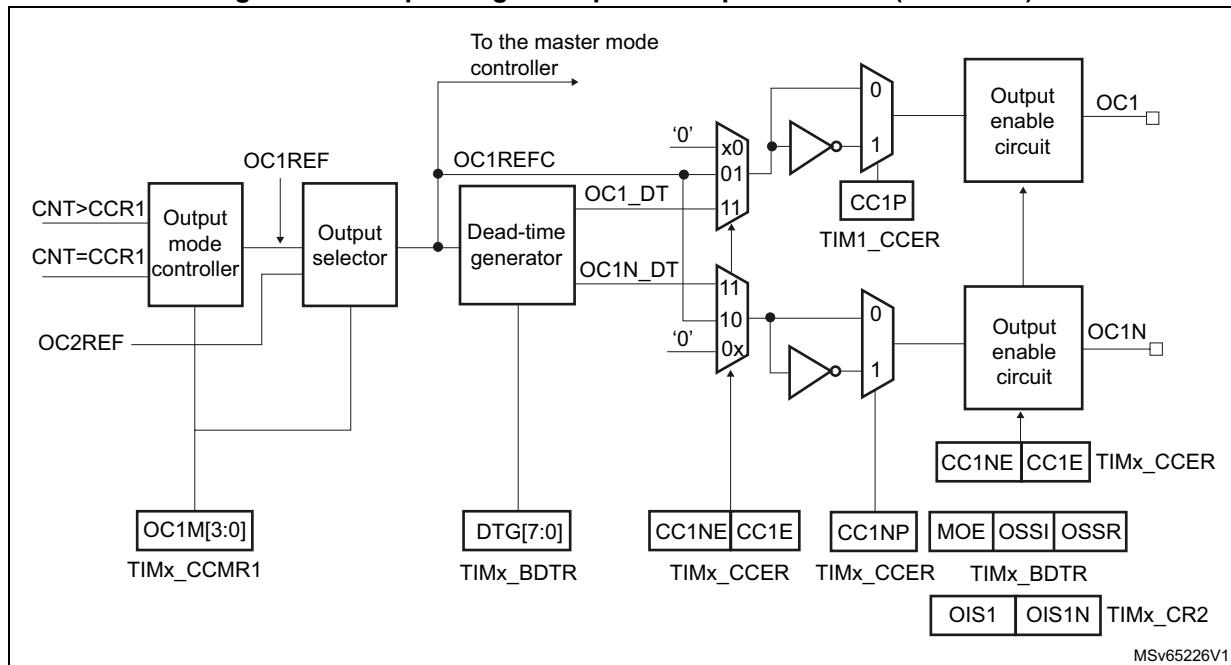


Figure 268. Output stage of capture/compare channel (channel 1)



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.

### 26.3.6 Input capture mode

In Input capture mode, the Capture/Compare registers (TIMx\_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx\_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx\_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx\_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx\_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at  $f_{DTS}$  frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx\_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx\_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx\_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

*Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx\_EGR register.*

### 26.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 26.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx\_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register.

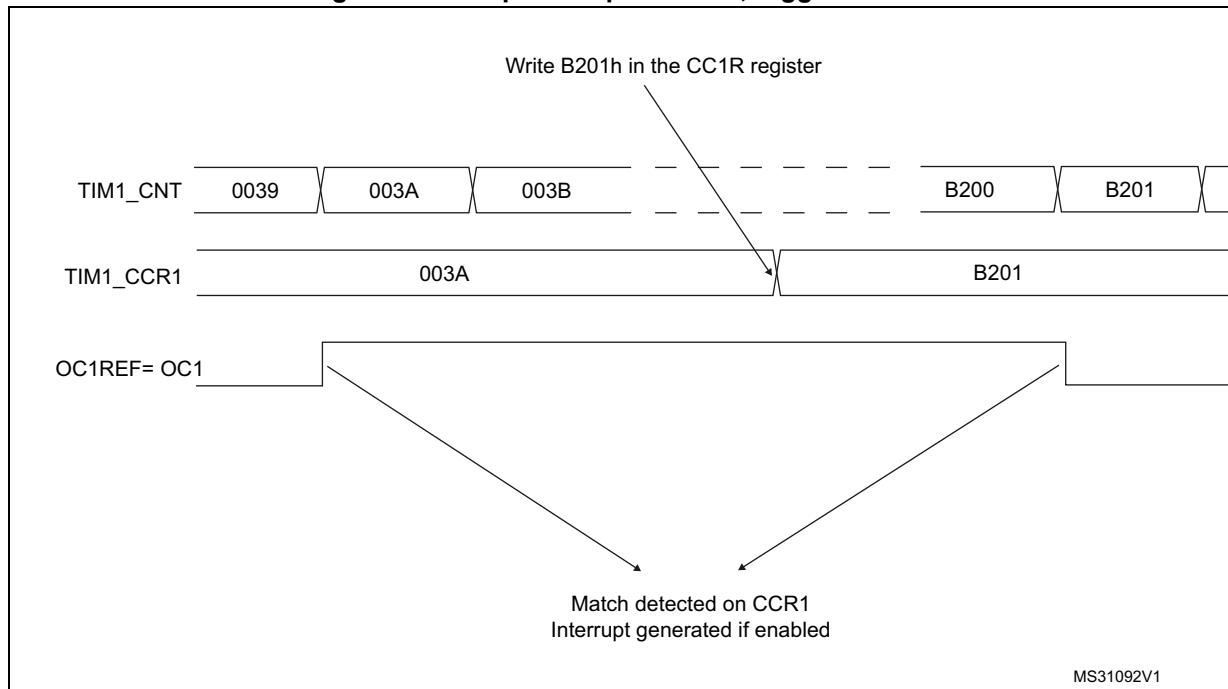
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

### Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
  - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
  - Write OCxPE = 0 to disable preload register
  - Write CCxP = 0 to select active high polarity
  - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 269](#).

Figure 269. Output compare mode, toggle on OC1



### 26.3.9 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx\_EGR register.

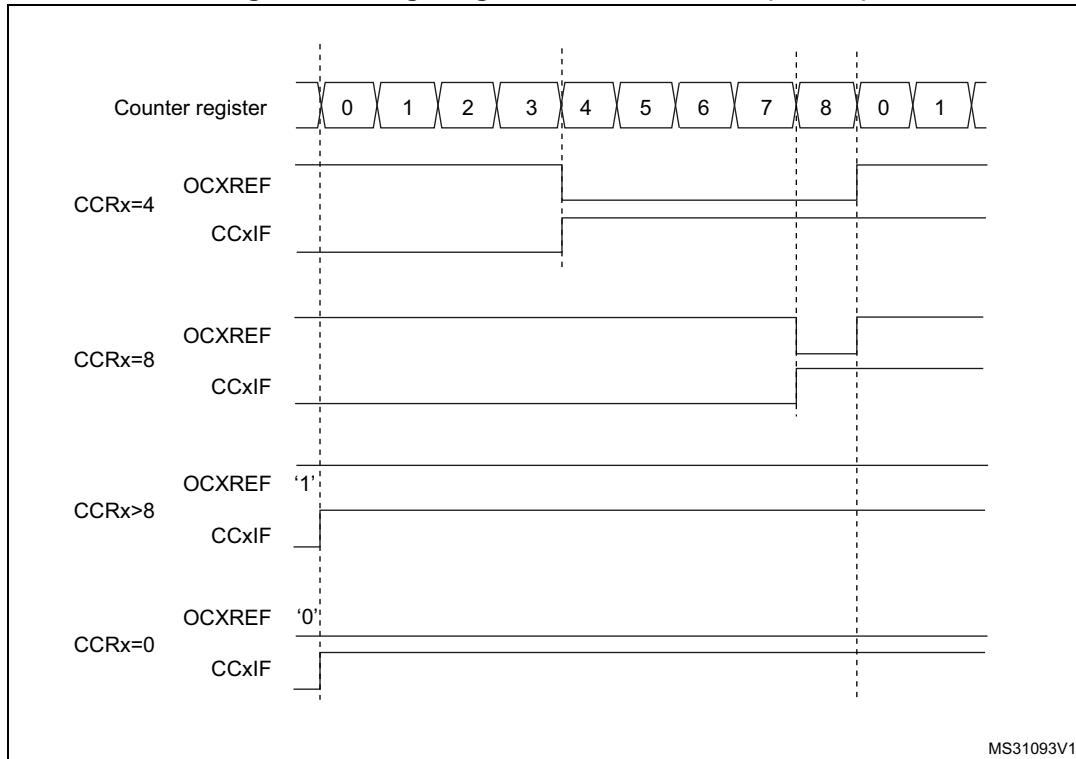
OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSS1 and OSSR bits (TIMx\_CCER and TIMx\_BDTR registers). Refer to the TIMx\_CCER register description for more details.

In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether TIMx\_CCRx  $\leq$  TIMx\_CNT or TIMx\_CNT  $\leq$  TIMx\_CCRx (depending on the direction of the counter).

The TIM16/TIM17 are capable of upcounting only. Refer to [Upcounting mode on page 837](#).

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 270](#) shows some edge-aligned PWM waveforms in an example where TIMx\_ARR=8.

Figure 270. Edge-aligned PWM waveforms (ARR=8)



### 26.3.10 Complementary outputs and dead-time insertion

The TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OC<sub>x</sub> or complementary OC<sub>xN</sub>) can be selected independently for each output. This is done by writing to the CC<sub>xP</sub> and CC<sub>xNP</sub> bits in the TIM<sub>x</sub>\_CCER register.

The complementary signals OC<sub>x</sub> and OC<sub>xN</sub> are activated by a combination of several control bits: the CC<sub>xE</sub> and CC<sub>xNE</sub> bits in the TIM<sub>x</sub>\_CCER register and the MOE, OIS<sub>x</sub>, OIS<sub>xN</sub>, OSS<sub>I</sub> and OSS<sub>R</sub> bits in the TIM<sub>x</sub>\_BDTR and TIM<sub>x</sub>\_CR2 registers. Refer to

*Table 170: Output control bits for complementary OC<sub>x</sub> and OC<sub>xN</sub> channels with break feature (TIM16/17) on page 872* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

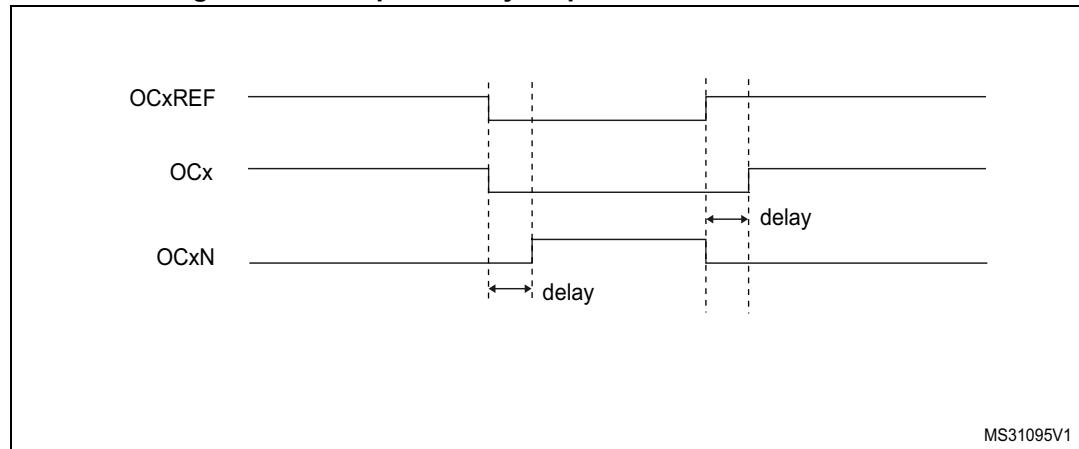
Dead-time insertion is enabled by setting both CC<sub>xE</sub> and CC<sub>xNE</sub> bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OC<sub>x</sub>REF, it generates 2 outputs OC<sub>x</sub> and OC<sub>xN</sub>. If OC<sub>x</sub> and OC<sub>xN</sub> are active high:

- The OC<sub>x</sub> output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OC<sub>xN</sub> output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

**Figure 271. Complementary output with dead-time insertion.**



**Figure 272. Dead-time waveforms with delay greater than the negative pulse.**

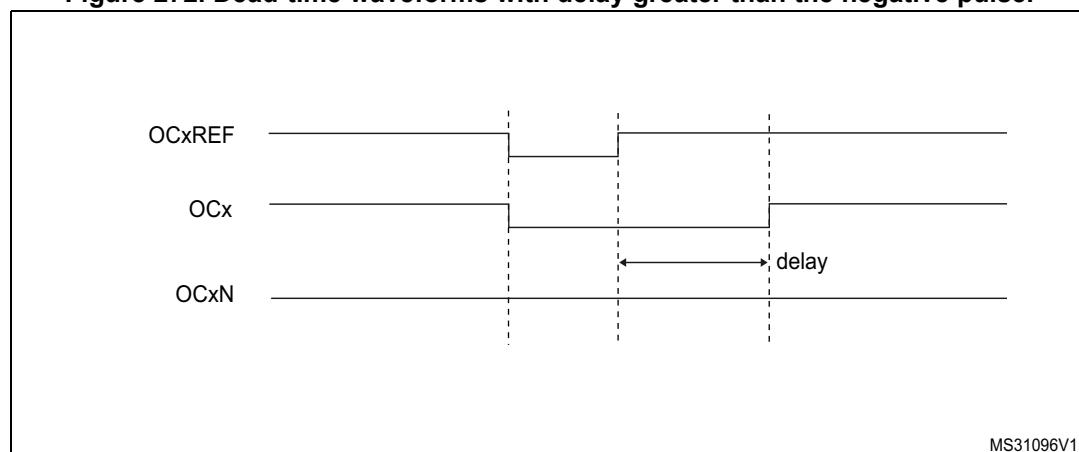
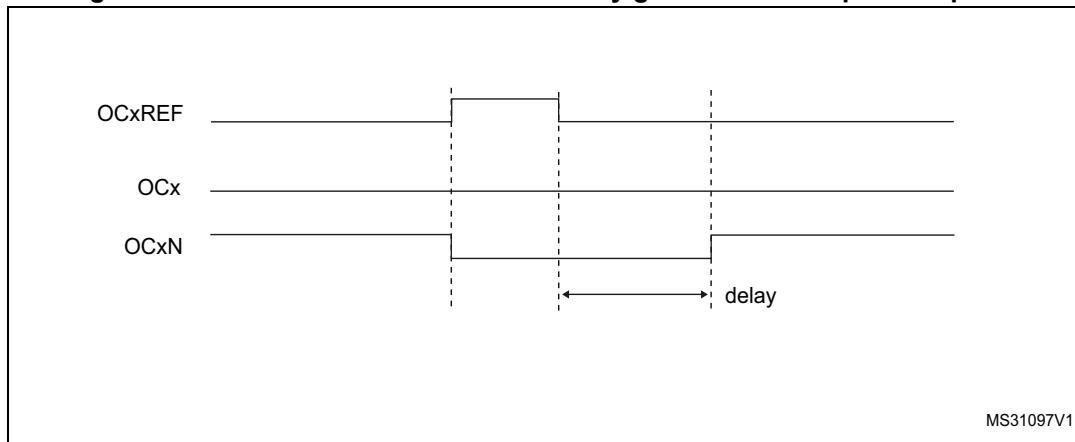


Figure 273. Dead-time waveforms with delay greater than the positive pulse.



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 26.4.14: TIMx break and dead-time register \(TIMx\\_BDTR\)\(x = 16 to 17\) on page 875](#) for delay calculation.

#### Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx\_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

**Note:** When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

#### 26.3.11 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM16/TIM17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx\_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx\_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay

must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO (selected by the OSS1 bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx\_CR2 register as soon as MOE=0. If OSS1=0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck\_tim clock cycles).
  - If OSS1=0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx\_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx\_DIER register is set.
- If the AOE bit in the TIMx\_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

**Note:** *If the MOE is reset by the CPU while the AOE bit is set, the outputs will be in idle state and forced to inactive level or Hi-Z depending on OSS1 value.*

*If both the MOE and AOE bits are reset by the CPU, the outputs will be in disabled state and driven with the level programmed in the OISx bit in the TIMx\_CR2 register.*

**Note:** *The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.*

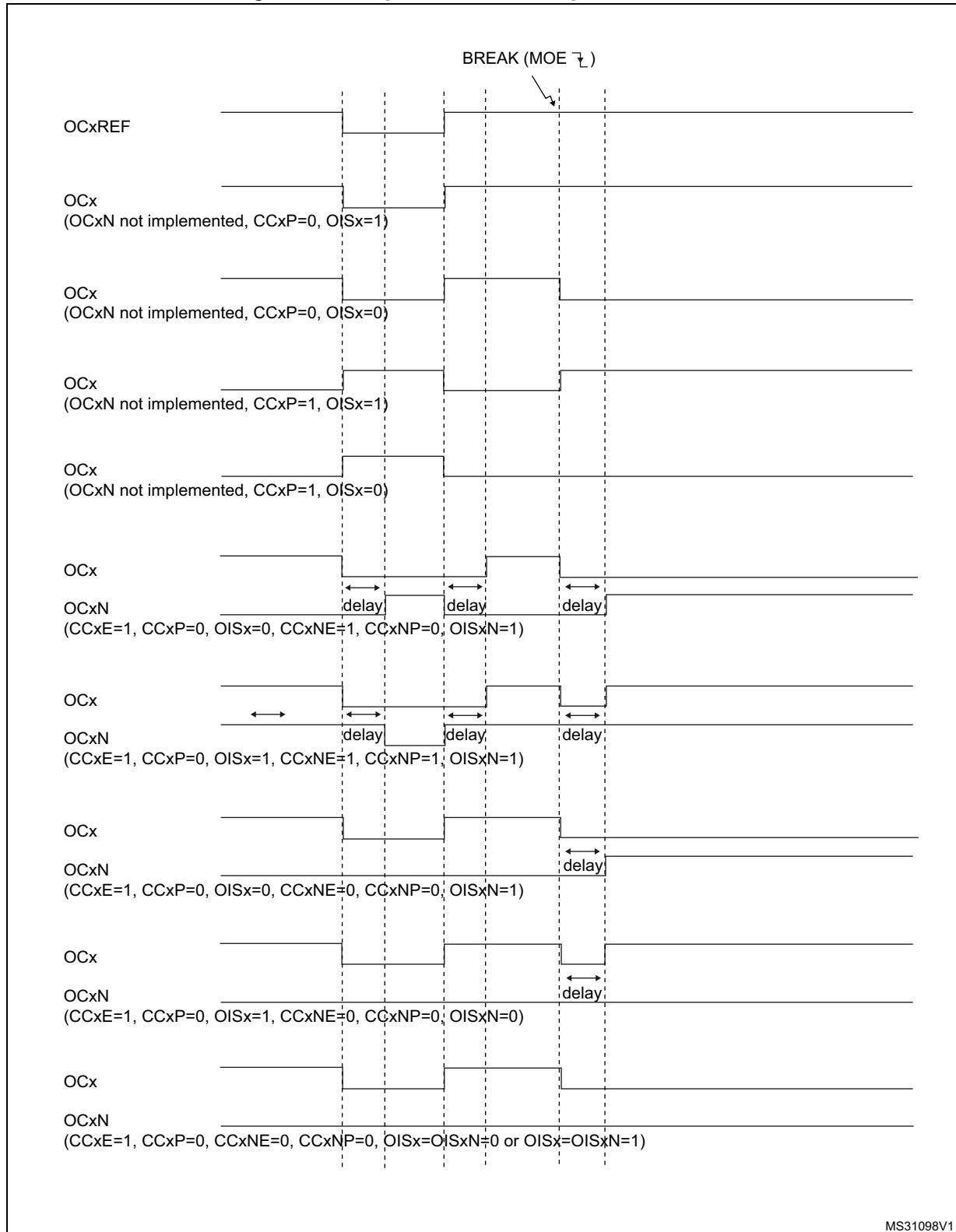
The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx\_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx\_BDTR register. Refer to

[Section 26.4.14: TIMx break and dead-time register \(TIMx\\_BDTR\)\(x = 16 to 17\) on page 875](#). The LOCK bits can be written only once after an MCU reset.

The [Figure 274](#) shows an example of behavior of the outputs in response to a break.

Figure 274. Output behavior in response to a break



MS31098V1

### 26.3.12 Bidirectional break inputs

The TIM16/TIM17 are featuring bidirectional break I/Os, as represented on [Figure 275](#).

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (BG) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see [Table 169](#))

**Table 169. Break protection disarming conditions**

MOE	BKDIR	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

#### Arming and re-arming break circuitry

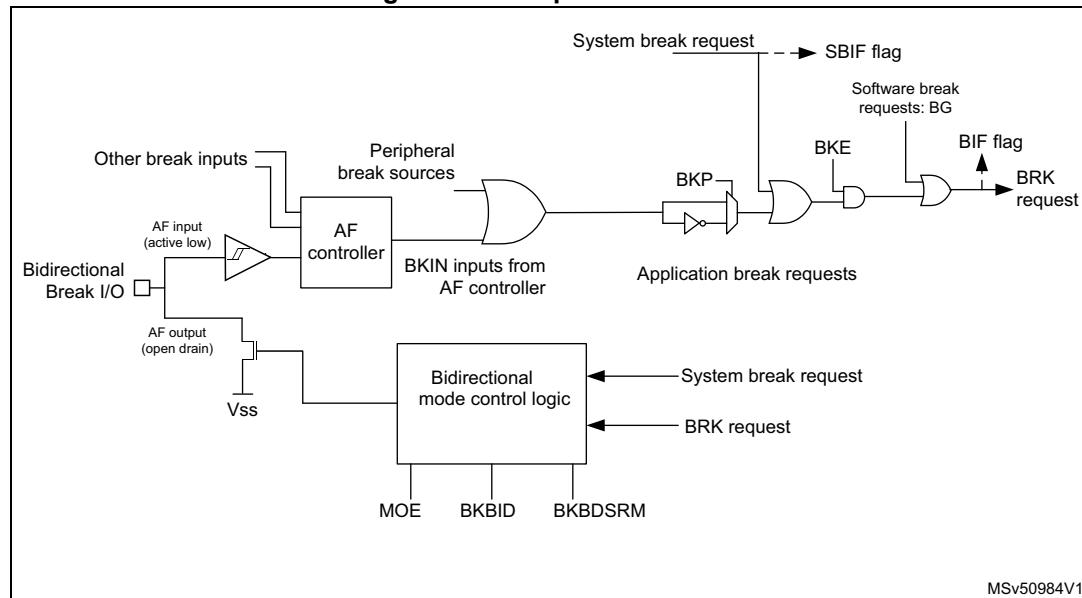
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

**Figure 275. Output redirection**



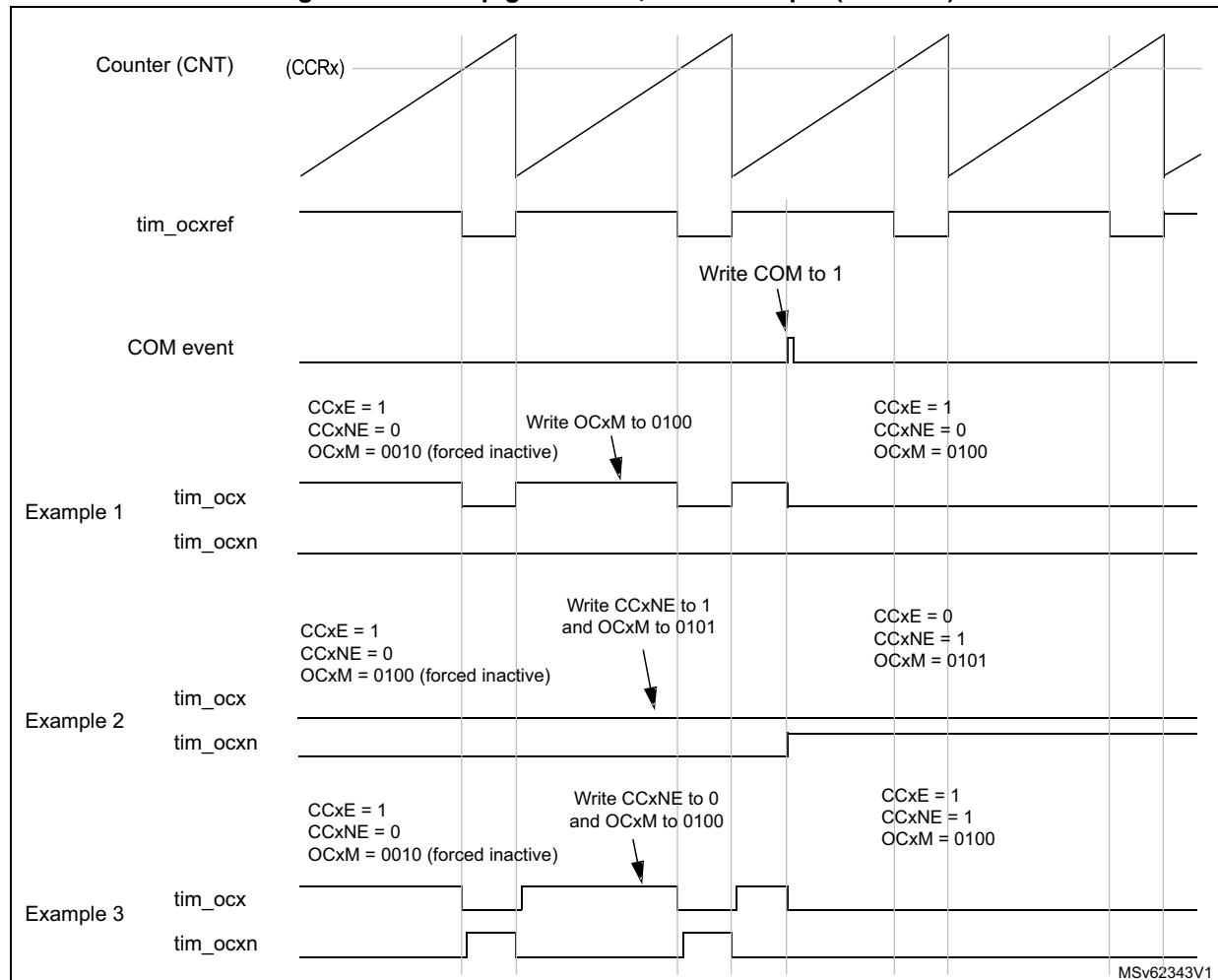
### 26.3.13 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx\_EGR register or by hardware (on tim\_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx\_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx\_DIER register) or a DMA request (if the COMDE bit is set in the TIMx\_DIER register).

The [Figure 276](#) describes the behavior of the tim\_ocx and tim\_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 276. 6-step generation, COM example (OSSR=1)



### 26.3.14 One-pulse mode

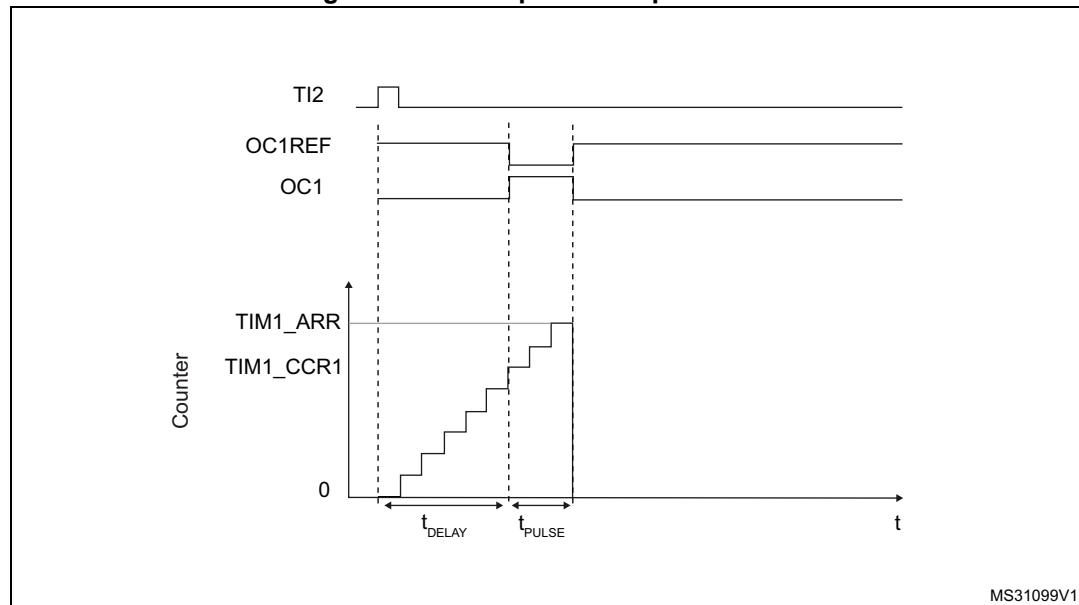
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $CNT < CCRx \leq ARR$  (in particular,  $0 < CCRx$ )

Figure 277. Example of one pulse mode



For example one may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx\_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx\_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='00110' in the TIMx\_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{DELAY}$  is defined by the value written in the `TIMx_CCR1` register.
- The  $t_{PULSE}$  is defined by the difference between the auto-reload value and the compare value (`TIMx_ARR` - `TIMx_CCR1`).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing `OC1M=111` in the `TIMx_CCMR1` register. Optionally the preload registers can be enabled by writing `OC1PE='1'` in the `TIMx_CCMR1` register and `ARPE` in the `TIMx_CR1` register. In this case one has to write the compare value in the `TIMx_CCR1` register, the auto-reload value in the `TIMx_ARR` register, generate an update by setting the `UG` bit and wait for external trigger event on `TI2`. `CC1P` is written to '0' in this example.

Since only 1 pulse is needed, a 1 must be written in the `OPM` bit in the `TIMx_CR1` register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: `OCx` fast enable

In One-pulse mode, the edge detection on `TIx` input set the `CEN` bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{DELAY}$  min we can get.

If one wants to output a waveform with the minimum delay, the `OCxFE` bit can be set in the `TIMx_CCMRx` register. Then `OCxRef` (and `OCx`) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. `OCxFE` acts only if the channel is configured in PWM1 or PWM2 mode.

### 26.3.15 UIF bit remapping

The `IUFREMAP` bit in the `TIMx_CR1` register forces a continuous copy of the Update Interrupt Flag `UIF` into bit 31 of the timer counter register (`TIMxCNT[31]`). This allows both the counter value and a potential roll-over condition signaled by the `UIFCPY` flag, to be atomically read. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the `UIF` and `UIFCPY` flags.

### 26.3.16 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (`TRGI`) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

### 26.3.17 DMA burst mode

The `TIMx` timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx\_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx\_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx\_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx\_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

Example:

00000: TIMx\_CR1,

00001: TIMx\_CR2,

00010: TIMx\_SMCR,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers ( $x = 2, 3, 4$ ) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
  - DMA channel peripheral address is the DMAR register address
  - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
  - Number of data to transfer = 3 (See note below).
  - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:  
DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

*Note:* A null value can be written to the reserved registers.

### 26.3.18 Using timer output as trigger for other timers (TIM16/TIM17)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx\_SMCR register on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer will detect the trigger.

For instance, if the destination's timer CK\_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

### 26.3.19 Debug mode

When the microcontroller enters debug mode (CPU1 Cortex<sup>®</sup>-M4 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 41.8.7: DBGMCU CPU1 APB2 peripheral freeze register \(DBGMCU\\_APB2FZR\)](#).

For safety purposes, when the counter is stopped (DBG\_TIMx\_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

## 26.4 TIM16/TIM17 registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

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

### 26.4.1 TIMx control register 1 (TIMx\_CR1)(x = 16 to 17)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping

- 0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.
- 1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK\_INT) frequency and the dead-time and sampling clock ( $t_{DTS}$ ) used by the dead-time generators and the digital filters (TIx),

- 00:  $t_{DTS} = t_{CK\_INT}$
- 01:  $t_{DTS} = 2 * t_{CK\_INT}$
- 10:  $t_{DTS} = 4 * t_{CK\_INT}$
- 11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

- 0: TIMx\_ARR register is not buffered
- 1: TIMx\_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode

- 0: Counter is not stopped at update event
- 1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

- 0: Any of the following events generate an update interrupt or DMA request if enabled.

These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

- 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

## 26.4.2 TIMx control register 2 (TIMx\_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

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

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

*Note: This bit acts only on channels that have a complementary output.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

*Note: This bit acts only on channels that have a complementary output.*

### 26.4.3 TIMx DMA/interrupt enable register (TIMx\_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable

0: CC1 DMA request disabled

1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled

1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable

0: Break interrupt disabled

1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE**: COM interrupt enable

0: COM interrupt disabled

1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled

1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

1: Update interrupt enabled

### 26.4.4 TIMx status register (TIMx\_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected

1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx\_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

**If channel CC1 is configured as output:** this flag is set when the content of the counter TIMx\_CNT matches the content of the TIMx\_CCR1 register. When the content of TIMx\_CCR1 is greater than the content of TIMx\_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx\_CR1 register for the full description.

**If channel CC1 is configured as input:** this bit is set when counter value has been captured in TIMx\_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx\_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx\_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx\_EGR register, if URS=0 and UDIS=0 in the TIMx\_CR1 register.

### 26.4.5 TIMx event generation register (TIMx\_EGR)(x = 16 to 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	BG	Res	COMG	Res	Res	Res	CC1G	UG							
								w		w				w	w

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

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

*Note: This bit acts only on channels that have a complementary output.*

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

**If channel CC1 is configured as output:**

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

**If channel CC1 is configured as input:**

The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

## 26.4.6 TIMx capture/compare mode register 1 [alternate] (TIMx\_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

### Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	IC1F[3:0]	IC1PSC[1:0]	CC1S[1:0]												
								rw	rw	rw	rw	rw	rw	rw	rw

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

#### Bits 7:4 IC1F[3:0]: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at  $f_{DTS}$
- 0001:  $f_{SAMPLING} = f_{CK\_INT}$ , N=4
- 0010:  $f_{SAMPLING} = f_{CK\_INT}$ , N=8
- 0011:  $f_{SAMPLING} = f_{CK\_INT}$ , N=8
- 0100:  $f_{SAMPLING} = f_{DTS}/2$ , N=
- 0101:  $f_{SAMPLING} = f_{DTS}/2$ , N=8
- 0110:  $f_{SAMPLING} = f_{DTS}/4$ , N=6
- 0111:  $f_{SAMPLING} = f_{DTS}/4$ , N=8
- 1000:  $f_{SAMPLING} = f_{DTS}/8$ , N=6
- 1001:  $f_{SAMPLING} = f_{DTS}/8$ , N=8
- 1010:  $f_{SAMPLING} = f_{DTS}/16$ , N=5
- 1011:  $f_{SAMPLING} = f_{DTS}/16$ , N=6
- 1100:  $f_{SAMPLING} = f_{DTS}/16$ , N=8
- 1101:  $f_{SAMPLING} = f_{DTS}/32$ , N=5
- 1110:  $f_{SAMPLING} = f_{DTS}/32$ , N=6
- 1111:  $f_{SAMPLING} = f_{DTS}/32$ , N=8

#### Bits 3:2 IC1PSC[1:0]: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).

The prescaler is reset as soon as CC1E='0' (TIMx\_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input.

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

Others: Reserved

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).*

#### 26.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx\_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

##### Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	rw
Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]										
									rw	rw	rw	rw	rw	rw	rw	rw

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

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

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

- 0000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs.
- 0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).
- 0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).
- 0011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1.
- 0100: Force inactive level - OC1REF is forced low.
- 0101: Force active level - OC1REF is forced high.
- 0110: PWM mode 1 - Channel 1 is active as long as TIMx\_CNT<TIMx\_CCR1 else inactive.
- 0111: PWM mode 2 - Channel 1 is inactive as long as TIMx\_CNT<TIMx\_CCR1 else active.
- All other values: Reserved

*Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

*In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.*

*The OC1M[3] bit is not contiguous, located in bit 16.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

- 0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime, the new value is taken in account immediately.
- 1: Preload register on TIMx\_CCR1 enabled. Read/Write operations access the preload register. TIMx\_CCR1 preload value is loaded in the active register at each update event.

*Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx\_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

- 0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
- 1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

Others: Reserved

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).*

### 26.4.8 TIMx capture/compare enable register (TIMx\_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CC1NP	CC1NE	CC1P	CC1E											

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

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

- 0: OC1N active high
- 1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S="00" (the channel is configured in output).*

*On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.*

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

- 0: Off - OC1N is not active. OC1N level is then function of MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.
- 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

- 0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
- 1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

**When CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

*On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 0 **CC1E**: Capture/Compare 1 output enable

- 0: Capture mode disabled / OC1 is not active (see below)
- 1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**When CC1 channel is configured as output**, the OC1 level depends on MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to [Table 170](#) for details.

**Table 170. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)**

Control bits					Output states <sup>(1)</sup>	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) OCx=0 OCxN=0, OCxN_EN=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	0		0	0		
	1		0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP	
	1		1	0	Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCx and OCxN both in active state	
	1		1	1		

- When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

**Note:** The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and GPIO control and alternate function registers.

#### 26.4.9 TIMx counter (TIMx\_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.														
r															
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx\_ISR register. If the UIFREMAP bit in TIMx\_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

#### 26.4.10 TIMx prescaler (TIMx\_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK\_CNT) is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

#### 26.4.11 TIMx auto-reload register (TIMx\_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 26.3.1: Time-base unit on page 835](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 26.4.12 TIMx repetition counter register (TIMx\_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

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

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

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP\_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP\_CNT is reloaded with REP value only at the repetition update event U\_RC, any write to the TIMx\_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

### 26.4.13 TIMx capture/compare register 1 (TIMx\_CCR1)(x = 16 to 17)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

**If channel CC1 is configured as output:**

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC1 output.

**If channel CC1 is configured as input:**

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

### 26.4.14 TIMx break and dead-time register (TIMx\_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
			rw		rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]						DTG[7:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

**Note:** As the BKBID, BKDSRM, AOE, BKP, BKE, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx\_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID:** Break Bidirectional

- 0: Break input BRK in input mode
- 1: Break input BRK in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM:** Break Disarm

- 0: Break input BRK is armed
- 1: Break input BRK is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bits 25:20 Reserved, must be kept at reset value.

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

Bit 15 **MOE:** Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

- 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSR bit.
- 1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx\_CCER register)

See OC/OCN enable description for more details ([Section 26.4.8: TIMx capture/compare enable register \(TIMx\\_CCER\)\(x = 16 to 17\) on page 870](#)).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled

1: Break inputs (BRK and CCS clock failure event) enabled

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details ([Section 26.4.8: TIMx capture/compare enable register \(TIMx\\_CCER\)\(x = 16 to 17\) on page 870](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details ([Section 26.4.8: TIMx capture/compare enable register \(TIMx\\_CCER\)\(x = 16 to 17\) on page 870](#)).

- 0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
- 1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx\_BDTR register, OISx and OISxN bits in TIMx\_CR2 register and BKE/BKP/AOE bits in TIMx\_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx\_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx\_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

*Note: The LOCK bits can be written only once after the reset. Once the TIMx\_BDTR register has been written, their content is frozen until the next reset.*

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx => DT = DTG[7:0] x t<sub>dtg</sub> with t<sub>dtg</sub> = t<sub>DTS</sub>

DTG[7:5] = 10x => DT = (64 + DTG[5:0]) x t<sub>dtg</sub> with t<sub>dtg</sub> = 2 x t<sub>DTS</sub>

DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t<sub>dtg</sub> with t<sub>dtg</sub> = 8 x t<sub>DTS</sub>

DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t<sub>dtg</sub> with t<sub>dtg</sub> = 16 x t<sub>DTS</sub>

Example if t<sub>DTS</sub> = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

*Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 26.4.15 TIMx DMA control register (TIMx\_DCR)(x = 16 to 17)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	DBL[4:0]						Res	Res	Res	DBA[4:0]			
			rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer,  
00001: 2 transfers,  
00010: 3 transfers,  
...  
10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

Example:

00000: TIMx\_CR1,  
00001: TIMx\_CR2,  
00010: TIMx\_SMCR,  
...

**Example:** Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx\_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx\_CR1 address.

#### 26.4.16 TIMx DMA address for full transfer (TIMx\_DMAR)(x = 16 to 17)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address  
(TIMx\_CR1 address) + (DBA + DMA index) x 4

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

### 26.4.17 TIM16 option register 1 (TIM16\_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TI1_RMP[1:0]														
															rw rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TI1\_RMP[1:0]**: Timer 16 input 1 connection

This bit is set and cleared by software.

00: TIM16 TI1 is connected to GPIO

01: TIM16 TI1 is connected to LSI

10: TIM16 TI1 is connected to LSE

11: TIM16 TI1 is connected to RTC wake-up interrupt

### 26.4.18 TIM16 alternate function register 1 (TIM16\_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BKCM P2P	BKCM P1P	BKINP	Res.	Res.	Res.	Res.	Res.	Res.	BKCM P2E	BKCM P1E	BKINE
				rw	rw	rw							rw	rw	rw

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

Bit 11 **BKCM2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input is active low

1: COMP2 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **BKCM1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input is active low

1: COMP1 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: BKIN input is active low
- 1: BKIN input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCM2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

- 0: COMP2 input disabled
- 1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 1 **BKCM1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

- 0: COMP1 input disabled
- 1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

- 0: BKIN input disabled
- 1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 26.4.19 TIM16 input selection register (TIM16\_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
												rw	rw	rw	rw
TI1SEL[3:0]															

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

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

- 0000: TIM16\_CH1 input
- Others: Reserved

### 26.4.20 TIM17 option register 1 (TIM17\_OR1)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TI1_RMP[1:0]														
															rw rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TI1\_RMP[1:0]**: Timer 17 input 1 connection

This bit is set and cleared by software.

00: TIM17 TI1 is connected to GPIO

01: TIM17 TI1 is connected to MSI

10: TIM17 TI1 is connected to HSE/32

11: TIM17 TI1 is connected to MCO

#### 26.4.21 TIM17 alternate function register 1 (TIM17\_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BKCM P2P	BKCM P1P	BKINP	Res.	Res.	Res.	Res.	Res.	Res.	BKCM P2E	BKCM P1E	BKINE
				rw	rw	rw							rw	rw	rw

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

Bit 11 **BKCM2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input is active low

1: COMP2 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **BKCM1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input is active low

1: COMP1 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: BKIN input is active low
- 1: BKIN input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:3 Reserved, must be kept at reset value.

Bit 2 **BKCM2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

- 0: COMP2 input disabled
- 1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 1 **BKCM1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

- 0: COMP1 input disabled
- 1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

- 0: BKIN input disabled
- 1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

## 26.4.22 TIM17 input selection register (TIM17\_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
												rw	rw	rw	rw
TI1SEL[3:0]															

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

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM17\_CH1 input

Others: Reserved

## 26.4.23 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 171. TIM16/TIM17 register map and reset values**

Table 171. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	<b>TIMx_RCR</b>	Res.	0	MOE	0	AOE	0	BKE	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	BKBDID	0	BKBDID	0	BKBDID	0	0	0	0	0	0	0	0	0	0															
0x34	<b>TIMx_CCR1</b>	Res.	0	BKDSRM	0	BKDSRM	0	BKDSRM	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	REP[7:0]	0	REP[7:0]	0	REP[7:0]	0	0	0	0	0	0	0	0	0	0															
0x44	<b>TIMx_BDTR</b>	Res.	0	DTG[7:0]	0	DTG[7:0]	0	DTG[7:0]	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	CCR1[15:0]	0	CCR1[15:0]	0	CCR1[15:0]	0	0	0	0	0	0	0	0	0	0															
0x48	<b>TIMx_DCR</b>	Res.	0	DBL[4:0]	0	DBL[4:0]	0	DBL[4:0]	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	DMAB[15:0]	0	DMAB[15:0]	0	DMAB[15:0]	0	0	0	0	0	0	0	0	0	0															
0x4C	<b>TIMx_DMAR</b>	Res.	0	LOC[1:0]	0	LOC[1:0]	0	LOC[1:0]	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	DTG[7:0]	0	DTG[7:0]	0	DTG[7:0]	0	0	0	0	0	0	0	0	0	0															
0x50	<b>TIM16_OR1</b>	Res.	0	DBA[4:0]	0	DBA[4:0]	0	DBA[4:0]	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	CCR1[15:0]	0	CCR1[15:0]	0	CCR1[15:0]	0	0	0	0	0	0	0	0	0	0															
0x50	<b>TIM17_OR1</b>	Res.	0	DTG[7:0]	0	DTG[7:0]	0	DTG[7:0]	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	CCR1[15:0]	0	CCR1[15:0]	0	CCR1[15:0]	0	0	0	0	0	0	0	0	0	0															
0x60	<b>TIM16_AF1</b>	Res.	0	DBL[4:0]	0	DBL[4:0]	0	DBL[4:0]	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	DMAB[15:0]	0	DMAB[15:0]	0	DMAB[15:0]	0	0	0	0	0	0	0	0	0	0															
0x68	<b>TIM16_TISEL</b>	Res.	0	LOC[1:0]	0	LOC[1:0]	0	LOC[1:0]	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	DTG[7:0]	0	DTG[7:0]	0	DTG[7:0]	0	0	0	0	0	0	0	0	0	0															
0x68	<b>TIM17_TISEL</b>	Res.	0	DBL[4:0]	0	DBL[4:0]	0	DBL[4:0]	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	CCR1[15:0]	0	CCR1[15:0]	0	CCR1[15:0]	0	0	0	0	0	0	0	0	0	0															

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 27 Low-power timer (LPTIM)

### 27.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. Also, the LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

### 27.2 LPTIM main features

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no embedded oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode

## 27.3 LPTIM implementation

*Table 172* and *Table 173* describe LPTIM implementation: the full set of features is implemented in LPTIM1. LPTIM2 supports a smaller set of features, but is otherwise identical to LPTIM1.

**Table 172. LPTIM features**

LPTIM modes/features <sup>(1)</sup>	LPTIM1	LPTIM2
Encoder mode	X	-
External input clock	X	X
Wake-up from Stop	(2)	(3)

1. X = supported.
2. Wake-up supported from Stop 0, Stop 1 and Stop 2 modes.
3. Wake-up supported from Stop 0 and Stop 1 modes.

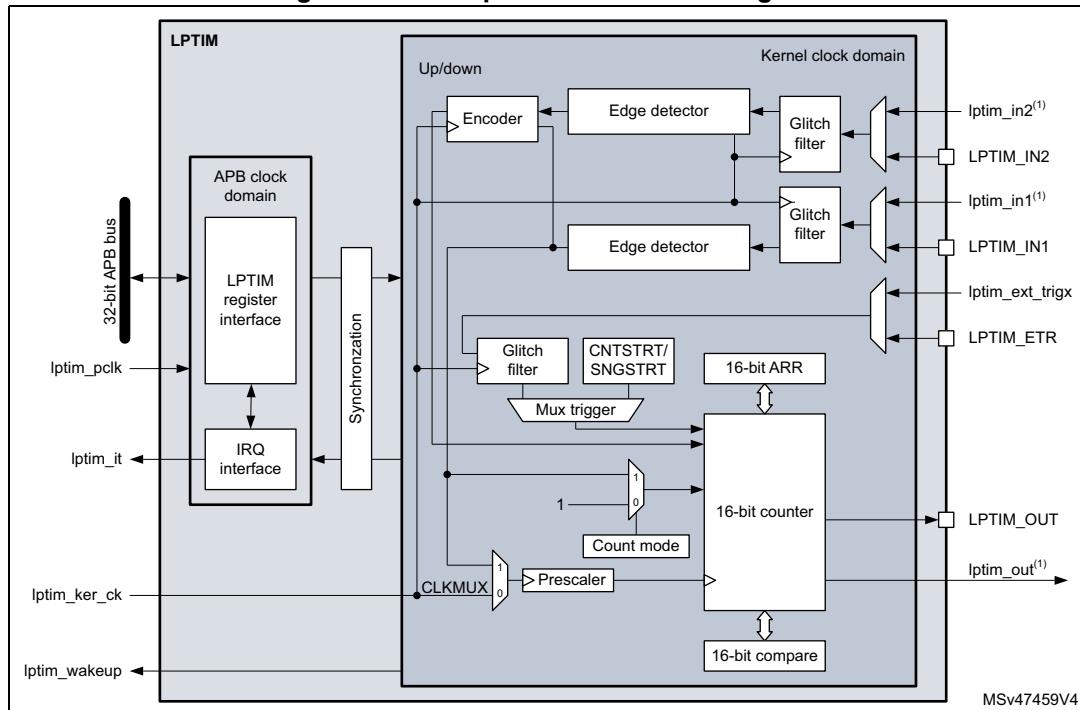
**Table 173. LPTIM implementation**

Feature	STM32WB55xx	STM32WB35xx
LPTIM1 external trigger	GPIO, RTC alarm A, RTC alarm B, RTC_TAMP[1:3]	GPIO, RTC alarm A, RTC alarm B, RTC_TAMP2,
LPTIM2 external trigger	COMP1_OUT, COMP2_OUT	COMP1_OUT, COMP2_OUT

## 27.4 LPTIM functional description

### 27.4.1 LPTIM block diagram

Figure 278. Low-power timer block diagram



1. *lptim\_in1* and *lptim\_in2* are respectively internal LPTIM input 1 and input 2 signals that can be connected to internal peripherals. *lptim\_out* is the internal LPTIM output signal that can be connected to internal peripherals.

### 27.4.2 LPTIM trigger mapping

The LPTIM external trigger connections are detailed hereafter:

Table 174. LPTIM1 external trigger connection

TRIGSEL	External trigger
<i>lptim_ext_trig0</i>	GPIO
<i>lptim_ext_trig1</i>	RTC alarm A
<i>lptim_ext_trig2</i>	RTC alarm B
<i>lptim_ext_trig3</i>	RTC_TAMP1 input detection
<i>lptim_ext_trig4</i>	RTC_TAMP2 input detection
<i>lptim_ext_trig5</i>	RTC_TAMP3 input detection
<i>lptim_ext_trig6</i>	COMP1_OUT
<i>lptim_ext_trig7</i>	COMP2_OUT

**Table 175. LPTIM2 external trigger connection**

TRIGSEL	External trigger
lptim_ext_trig0	GPIO
lptim_ext_trig1	RTC alarm A
lptim_ext_trig2	RTC alarm B
lptim_ext_trig3	RTC_TAMP1 input detection
lptim_ext_trig4	RTC_TAMP2 input detection
lptim_ext_trig5	RTC_TAMP3 input detection
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT

### 27.4.3 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or Pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM will use an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal should also be provided (first configuration). In this case, the internal clock signal frequency should be at least four times higher than the external clock signal frequency.

### 27.4.4 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source should first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

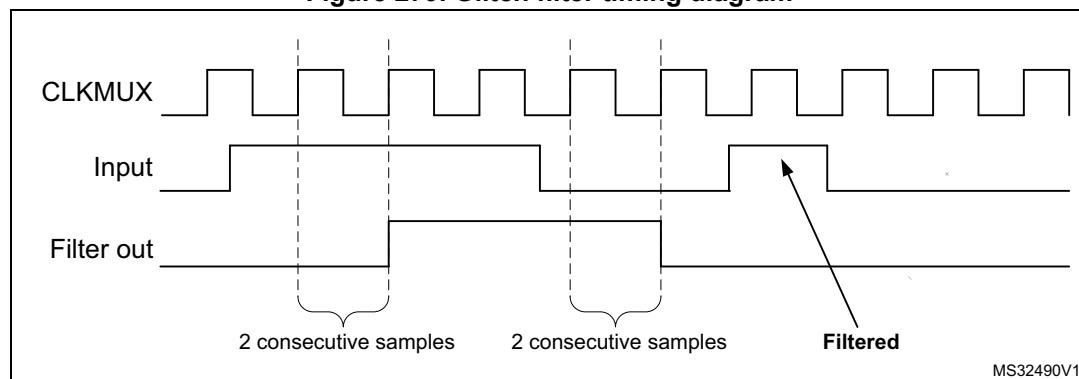
The digital filters are divided into two groups:

- The first group of digital filters protects the LPTIM external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.

**Note:** *The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.*

The filter sensitivity acts on the number of consecutive equal samples that should be detected on one of the LPTIM inputs to consider a signal level change as a valid transition. [Figure 279](#) shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

**Figure 279. Glitch filter timing diagram**



**Note:** *In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT and TRGFLT bits to '0'. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.*

#### 27.4.5 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] 3-bit field. The table below lists all the possible division ratios:

**Table 176. Prescaler division ratios**

programming	dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

### 27.4.6 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals '00', The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than '00', TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it will be ignored (unless timeout function is enabled).

**Note:** *The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled will be discarded by hardware.*

**Note:** *When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM\_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.*

### 27.4.7 Operating mode

The LPTIM features two operating modes:

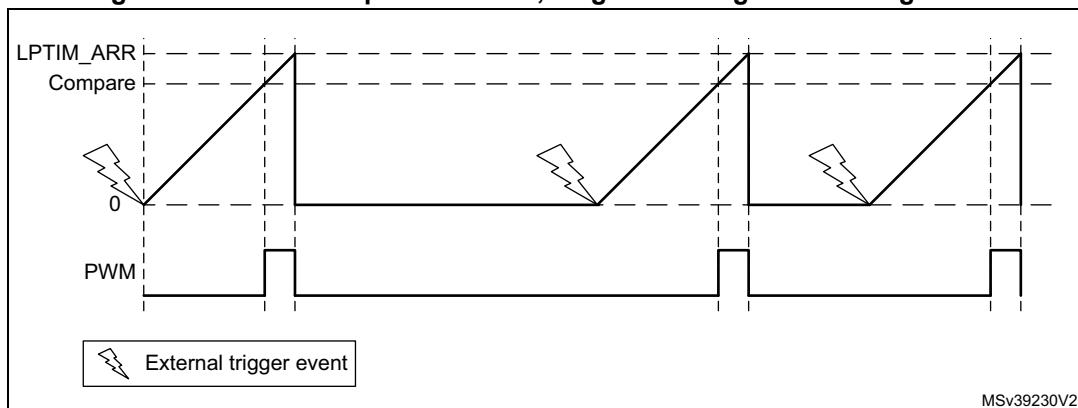
- The Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when reaching the ARR value.

#### One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

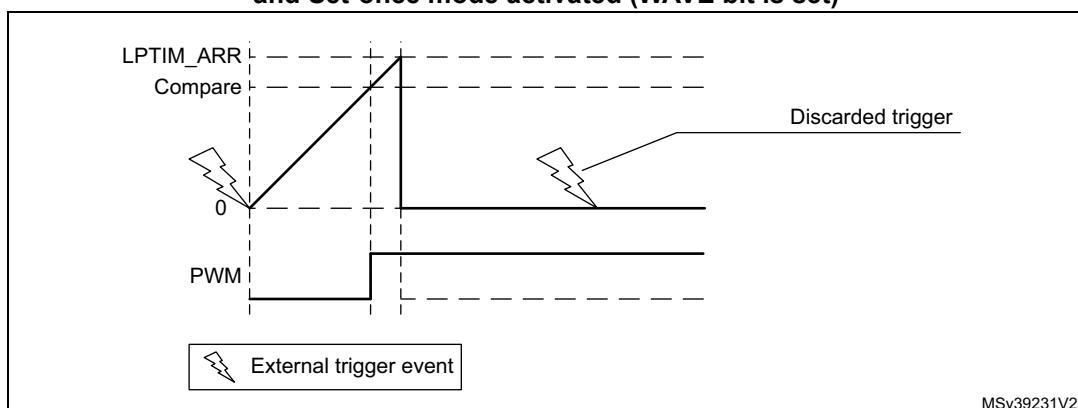
A new trigger event will re-start the timer. Any trigger event occurring after the counter starts and before the counter reaches ARR will be discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the counter register has stopped (contains zero value), will start the counter for a new one-shot counting cycle as shown in [Figure 280](#).

**Figure 280. LPTIM output waveform, single counting mode configuration**

Set-once mode activated:

It should be noted that when the WAVE bit-field in the LPTIM\_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in [Figure 281](#).

**Figure 281. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)**

In case of software start (TRIGEN[1:0] = '00'), the SNGSTRT setting will start the counter for one-shot counting.

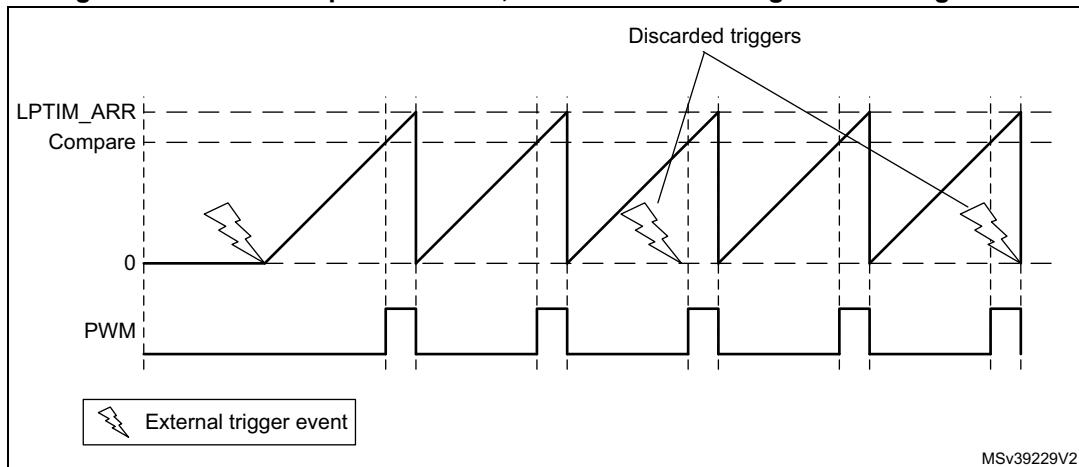
### Continuous mode

To enable the continuous counting, the CNTSTART bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTART is set will start the counter for continuous counting. Any subsequent external trigger event will be discarded as shown in [Figure 282](#).

In case of software start (TRIGEN[1:0] = '00'), setting CNTSTART will start the counter for continuous counting.

Figure 282. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (The ENABLE bit is set to '1'). It is possible to change "on the fly" from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT will switch the LPTIM to the One-shot mode. The counter (if active) will stop as soon as it reaches ARR.

If the One-shot mode was previously selected, setting CNTSTRT will switch the LPTIM to the Continuous mode. The counter (if active) will restart as soon as it reaches ARR.

#### 27.4.8 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMEOUT bit.

The first trigger event will start the timer, any successive trigger event will reset the counter and the timer will restart.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

#### 27.4.9 Waveform generation

Two 16-bit registers, the LPTIM\_ARR (autoreload register) and LPTIM\_CMP (compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM\_CNT exceeds the compare value in LPTIM\_CMP. The LPTIM output is reset as soon as a match occurs between the LPTIM\_ARR and the LPTIM\_CNT registers.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM\_ARR register value be strictly greater than the LPTIM\_CMP register value.

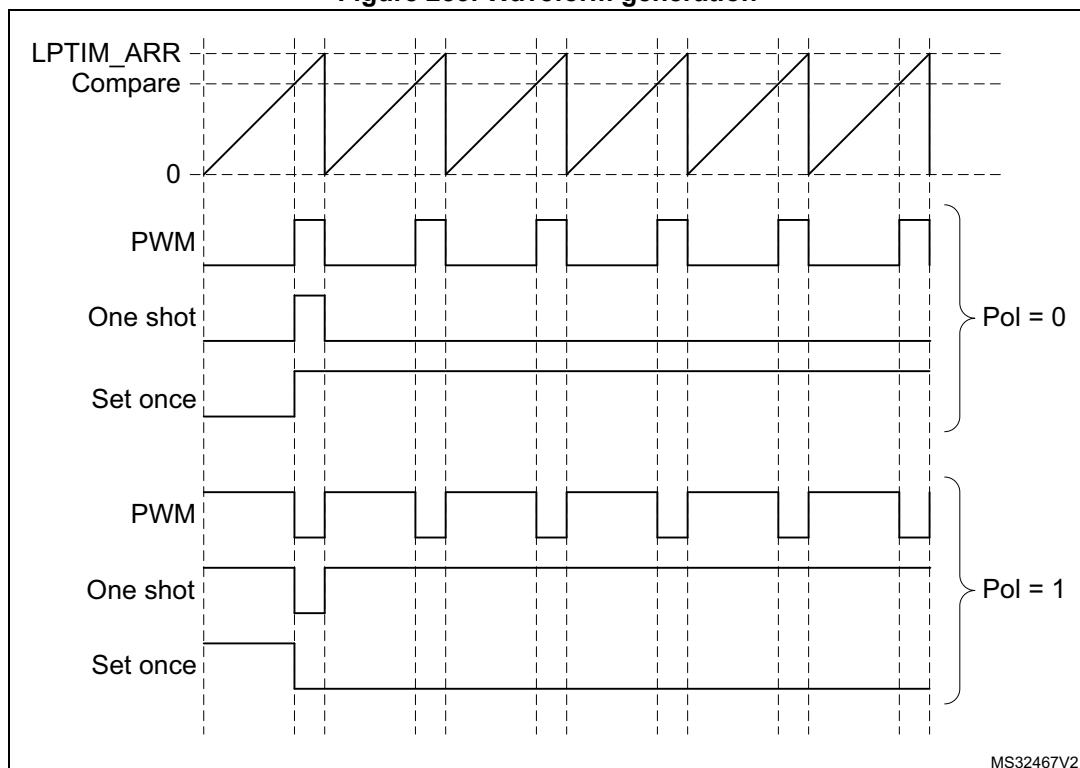
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to '0' forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTART or SNGSTART.
- Setting the WAVE bit to '1' forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value will change immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. [Figure 283](#) below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL bit.

**Figure 283. Waveform generation**



#### 27.4.10 Register update

The LPTIM\_ARR register and LPTIM\_CMP register are updated immediately after the APB bus write operation, or at the end of the current period if the timer is already started.

The PRELOAD bit controls how the LPTIM\_ARR and the LPTIM\_CMP registers are updated:

- When the PRELOAD bit is reset to '0', the LPTIM\_ARR and the LPTIM\_CMP registers are immediately updated after any write access.
- When the PRELOAD bit is set to '1', the LPTIM\_ARR and the LPTIM\_CMP registers are updated at the end of the current period, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the

counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag and the CMPOK flag in the LPTIM\_ISR register indicate when the write operation is completed to respectively the LPTIM\_ARR register and the LPTIM\_CMP register.

After a write to the LPTIM\_ARR register or the LPTIM\_CMP register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag or the CMPOK flag be set, will lead to unpredictable results.

#### 27.4.11 Counter mode

The LPTIM counter can be used to count external events on the LPTIM Input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source will be used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0  
The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1  
The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal should never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).

- CKSEL = 1: the LPTIM is clocked by an external clock source  
COUNTMODE value is don't care.

In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

#### 27.4.12 Timer enable

The ENABLE bit located in the LPTIM\_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM is actually enabled.

The LPTIM\_CFGR and LPTIM\_IER registers must be modified only when the LPTIM is disabled.

#### 27.4.13 Timer counter reset

In order to reset the content of LPTIM\_CNT register to zero, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM\_CR register. After setting the COUNTRST bit-field to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic will elapse before the reset is taken into account. This will make the LPTIM counter count few extra pluses between the time when the reset is trigger and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.
- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM\_CR register. When this bit is set to '1', any read access to the LPTIM\_CNT register will reset its content to zero. Asynchronous reset should be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset should be applied only when there is enough insurance that no toggle will occur on the LPTIM Input1.

It should be noted that to read reliably the content of the LPTIM\_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM\_CNT register.

---

**Warning:** **There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. So developer should make sure that these two mechanisms are used exclusively.**

---

#### 27.4.14 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM\_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM\_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two Down and Up flags in the LPTIM\_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to '1'. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

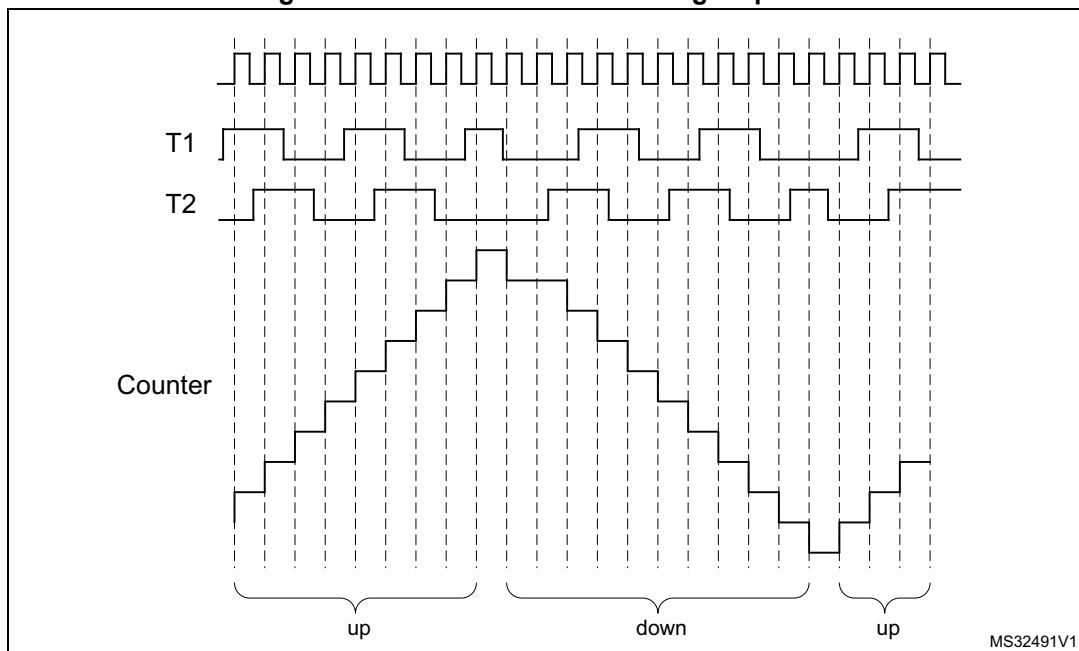
Table 177. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

**Caution:** In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to '0'. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be '000').

Figure 284. Encoder mode counting sequence



#### 27.4.15 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the `DBG_LPTIM_STOP` configuration bit in the DBG module.

### 27.5 LPTIM low-power modes

Table 178. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode (refer to <a href="#">Section 27.3: LPTIM implementation</a> ).
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

## 27.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM\_IER register:

- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable (up / down / both).

**Note:** *If any bit in the LPTIM\_IER register (Interrupt Enable Register) is set after that its corresponding flag in the LPTIM\_ISR register (Status Register) is set, the interrupt is not asserted.*

**Table 179. Interrupt events**

Interrupt event	Description
Compare match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the compare register (LPTIM_CMP).
Auto-reload match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the Auto-reload register (LPTIM_ARR).
External trigger event	Interrupt flag is raised when an external trigger event is detected
Auto-reload register update OK	Interrupt flag is raised when the write operation to the LPTIM_ARR register is complete.
Compare register update OK	Interrupt flag is raised when the write operation to the LPTIM_CMP register is complete.
Direction change	Used in Encoder mode. Two interrupt flags are embedded to signal direction change: – UP flag signals up-counting direction change – DOWN flag signals down-counting direction change.

## 27.7 LPTIM registers

The peripheral registers can only be accessed by words (32-bit).

### 27.7.1 LPTIM interrupt and status register (LPTIM\_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DOWN	UP	ARR OK	CMP OK	EXT TRIG	ARRM	CMPM								
								r	r	r	r	r	r	r	r

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

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM\_ICR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to [Section 27.3: LPTIM implementation](#).*

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM\_ICR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to [Section 27.3: LPTIM implementation](#).*

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM\_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM\_ICR register.

Bit 3 **CMPOK**: Compare register update OK

CMPOK is set by hardware to inform application that the APB bus write operation to the LPTIM\_CMP register has been successfully completed. CMPOK flag can be cleared by writing 1 to the CMPOKCF bit in the LPTIM\_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM\_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM\_CNT register's value reached the LPTIM\_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM\_ICR register.

Bit 0 **CMPM**: Compare match

The CMPM bit is set by hardware to inform application that LPTIM\_CNT register value reached the LPTIM\_CMP register's value. CMPM flag can be cleared by writing 1 to the CMPMCF bit in the LPTIM\_ICR register.

## 27.7.2 LPTIM interrupt clear register (LPTIM\_ICR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DOWNCF	UPCF	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF								
									w	w	w	w	w	w	w

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

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM\_ISR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 27.3: LPTIM implementation.*

Bit 5 **UPCF**: Direction change to UP clear flag

Writing 1 to this bit clear the UP flag in the LPTIM\_ISR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 27.3: LPTIM implementation.*

Bit 4 **ARROKCF**: Autoreload register update OK clear flag

Writing 1 to this bit clears the ARROK flag in the LPTIM\_ISR register

Bit 3 **CMPOKCF**: Compare register update OK clear flag

Writing 1 to this bit clears the CMPOK flag in the LPTIM\_ISR register

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag

Writing 1 to this bit clears the EXTTRIG flag in the LPTIM\_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag

Writing 1 to this bit clears the ARRM flag in the LPTIM\_ISR register

Bit 0 **CMPMCF**: Compare match clear flag

Writing 1 to this bit clears the CMPM flag in the LPTIM\_ISR register

## 27.7.3 LPTIM interrupt enable register (LPTIM\_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DOWNIE	UPIE	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE								
									rw	rw	rw	rw	rw	rw	rw

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

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable

- 0: DOWN interrupt disabled
- 1: DOWN interrupt enabled

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 27.3: LPTIM implementation.*

Bit 5 **UPIE**: Direction change to UP Interrupt Enable

- 0: UP interrupt disabled
- 1: UP interrupt enabled

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 27.3: LPTIM implementation.*

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable

- 0: ARROK interrupt disabled
- 1: ARROK interrupt enabled

Bit 3 **CMPOKIE**: Compare register update OK Interrupt Enable

- 0: CMPOK interrupt disabled
- 1: CMPOK interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable

- 0: EXTTRIG interrupt disabled
- 1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable

- 0: ARRM interrupt disabled
- 1: ARRM interrupt enabled

Bit 0 **CMPMIE**: Compare match Interrupt Enable

- 0: CMPM interrupt disabled
- 1: CMPM interrupt enabled

**Caution:** The LPTIM\_IER register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0')

#### 27.7.4 LPTIM configuration register (LPTIM\_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRELOAD	WAVPOL	WAVE	TIMOUT	TRIGEN[1:0]	Res.	
							rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

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

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable

The ENC bit controls the Encoder mode

- 0: Encoder mode disabled
- 1: Encoder mode enabled

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 27.3: LPTIM implementation.*

Bit 23 **COUNTMODE**: counter mode enabled

The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:

- 0: the counter is incremented following each internal clock pulse
- 1: the counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode

The PRELOAD bit controls the LPTIM\_ARR and the LPTIM\_CMP registers update modality

- 0: Registers are updated after each APB bus write access
- 1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity

The WAVEPOL bit controls the output polarity

- 0: The LPTIM output reflects the compare results between LPTIM\_CNT and LPTIM\_CMP registers
- 1: The LPTIM output reflects the inverse of the compare results between LPTIM\_CNT and LPTIM\_CMP registers

Bit 20 **WAVE**: Waveform shape

The WAVE bit controls the output shape

- 0: Deactivate Set-once mode
- 1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable

The TIMOUT bit controls the Timeout feature

- 0: A trigger event arriving when the timer is already started will be ignored
- 1: A trigger event arriving when the timer is already started will reset and restart the counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity

The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:

- 00: software trigger (counting start is initiated by software)
- 01: rising edge is the active edge
- 10: falling edge is the active edge
- 11: both edges are active edges

## Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that will serve as a trigger event for the LPTIM among the below 8 available sources:

- 000: lptim\_ext\_trig0
- 001: lptim\_ext\_trig1
- 010: lptim\_ext\_trig2
- 011: lptim\_ext\_trig3
- 100: lptim\_ext\_trig4
- 101: lptim\_ext\_trig5
- 110: lptim\_ext\_trig6
- 111: lptim\_ext\_trig7

See [Section 27.4.2: LPTIM trigger mapping](#) for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]**: Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

- 000: /1
- 001: /2
- 010: /4
- 011: /8
- 100: /16
- 101: /32
- 110: /64
- 111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFILT[1:0]**: Configurable digital filter for trigger

The TRGFILT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

- 00: any trigger active level change is considered as a valid trigger
- 01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
- 10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
- 11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]**: Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any external clock signal level change is considered as a valid transition

01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]**: Clock polarity

If LPTIM is clocked by an external clock source:

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00:the rising edge is the active edge used for counting.

If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 1 is active.

01:the falling edge is the active edge used for counting

If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 2 is active.

10:both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.

If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 3 is active.

11:not allowed

Refer to [Section 27.4.14: Encoder mode](#) for more details about Encoder mode sub-modes.

Bit 0 **CKSEL**: Clock selector

The CKSEL bit selects which clock source the LPTIM will use:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

**Caution:** The LPTIM\_CFG register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0').

### 27.7.5 LPTIM control register (LPTIM\_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RST ARE	COUN TRST	CNT STRT	SNG STRT	ENABE										

Bits 31:5 Reserved, must be kept at reset value.

**Bit 4 RSTARE: Reset after read enable**

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM\_CNT register will asynchronously reset LPTIM\_CNT register content.

This bit can be set only when the LPTIM is enabled.

**Caution:** This bitfield is write-only. This means that the bit cannot be read back to verify the value which has been written. As an example, if this bit is set to 1, attempting to read it back will return 0 even if the "Reset after read" function is enabled (due to the fact that this bitfield has previously been written to 1). To turn off the "Reset after read" or to make sure that it has already been turned off, this bit should be reset (by programming it to 0) even if it already contains 0.

**Bit 3 COUNTRST: Counter reset**

This bit is set by software and cleared by hardware. When set to '1' this bit will trigger a synchronous reset of the LPTIM\_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

**Caution:** COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software should consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

**Bit 2 CNTSTRT: Timer start in Continuous mode**

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = '00'), setting this bit starts the LPTIM in Continuous mode. If the software start is disabled (TRIGEN[1:0] different than '00'), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer will not stop at the next match between the LPTIM\_ARR and LPTIM\_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It will be automatically reset by hardware.

**Bit 1 SNGSTRT: LPTIM start in Single mode**

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = '00'), setting this bit starts the LPTIM in single pulse mode. If the software start is disabled (TRIGEN[1:0] different than '00'), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM will stop at the following match between LPTIM\_ARR and LPTIM\_CNT registers.

This bit can only be set when the LPTIM is enabled. It will be automatically reset by hardware.

**Bit 0 ENABLE: LPTIM enable**

The ENABLE bit is set and cleared by software.

0:LPTIM is disabled

1:LPTIM is enabled

## 27.7.6 LPTIM compare register (LPTIM\_CMP)

Address offset: 0x014

Reset value: 0x0000 0000

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

CMP is the compare value used by the LPTIM.

**Caution:** The LPTIM\_CMP register must only be modified when the LPTIM is enabled (ENABLE bit set to '1').

## 27.7.7 LPTIM autoreload register (LPTIM\_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

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

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

Bits 15:0 **ARR[15:0]**: Auto reload value

ARR is the autoreload value for the LPTIM.

This value must be strictly greater than the CMP[15:0] value.

**Caution:** The LPTIM\_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to '1').

## 27.7.8 LPTIM counter register (LPTIM\_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **CNT[15:0]**: Counter value

When the LPTIM is running with an asynchronous clock, reading the LPTIM\_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical.

It should be noted that for a reliable LPTIM\_CNT register read access, two consecutive read accesses must be performed and compared. A read access can be considered reliable when the values of the two consecutive read accesses are equal.

## 27.7.9 LPTIM1 option register (LPTIM1\_OR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OR_1	OR_0													
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OR\_1**: Option register bit 1

- 0: LPTIM1 input 2 is connected to I/O
- 1: LPTIM1 input 2 is connected to COMP2\_OUT

Bit 0 **OR\_0**: Option register bit 0

- 0: LPTIM1 input 1 is connected to I/O
- 1: LPTIM1 input 1 is connected to COMP1\_OUT

### 27.7.10 LPTIM2 option register (LPTIM2\_OR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OR_1	OR_0													
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OR\_1**: Option register bit 1

- 0: LPTIM2 input 1 is connected to I/O
- 1: LPTIM2 input 1 is connected to COMP2\_OUT

Bit 0 **OR\_0**: Option register bit 0

- 0: LPTIM2 input 1 is connected to I/O
- 1: LPTIM2 input 1 is connected to COMP1\_OUT

### 27.7.11 LPTIM register map

The following table summarizes the LPTIM registers.

Table 180. LPTIM register map and reset values

Offset	Register name	Reset value
0x000	LPTIM_ISR	Res. 31
	Reset value	Res. 30
0x004	LPTIM_ICR	Res. 29
	Reset value	Res. 28
0x008	LPTIM_IER	Res. 27
	Reset value	Res. 26
0x00C	LPTIM_CFGR	Res. 25
	Reset value	Res. 24
0x010	LPTIM_CR	Res. 23
	Reset value	Res. 22
0x014	LPTIM_CMP	Res. 21
	Reset value	Res. 20
0x018	LPTIM_ARR	Res. 19
	Reset value	Res. 18
0x01C	LPTIM_CNT	Res. 17
	Reset value	Res. 16
0x020	LPTIM_OR	Res. 15
	Reset value	Res. 14

1. If LPTIM does not support encoder mode feature, this bit is reserved. Please refer to [Section 27.3: LPTIM implementation](#).

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 28 Infrared interface (IRTIM)

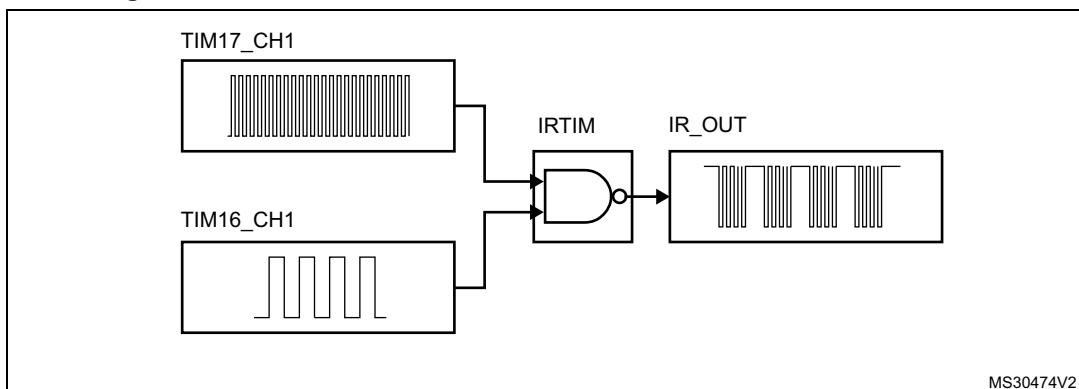
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with TIM16 and TIM17 as shown in [Figure 285](#).

To generate the infrared remote control signals, the IR interface must be enabled and TIM16 channel 1 (TIM16\_OC1) and TIM17 channel 1 (TIM17\_OC1) must be properly configured to generate correct waveforms.

The infrared receiver can be implemented easily through a basic input capture mode.

**Figure 285. IRTIM internal hardware connections with TIM16 and TIM17**



All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 generates the modulation envelope.

The infrared function is output on the IR\_OUT pin. The activation of this function is done through the GPIOx\_AFRx register by enabling the related alternate function bit.

The high sink LED driver capability (only available on the PB9 pin) can be activated through the I2C\_PB9\_FMP bit in the SYSCFG\_CFGR1 register and used to sink the high current needed to directly control an infrared LED.

## 29 Real-time clock (RTC)

### 29.1 Introduction

The RTC provides an automatic wake-up to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

The RTC includes also a periodic programmable wake-up flag with interrupt capability.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.

Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After Backup domain reset, all RTC registers are protected against possible parasitic write accesses.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

## 29.2 RTC main features

The RTC unit main features are the following (see [Figure 286: RTC block diagram](#)):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Programmable alarm with interrupt function. The alarm can be triggered by any combination of the calendar fields.
- Automatic wake-up unit generating a periodic flag that triggers an automatic wake-up interrupt.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Accurate synchronization with an external clock using the subsecond shift feature.
- Digital calibration circuit (periodic counter correction): 0.95 ppm accuracy, obtained in a calibration window of several seconds
- Time-stamp function for event saving
- Tamper detection event with configurable filter and internal pull-up
- Maskable interrupts/events:
  - Alarm A
  - Alarm B
  - Wake-up interrupt
  - Time-stamp
  - Tamper detection
- 20 backup registers.

## 29.3 RTC implementation

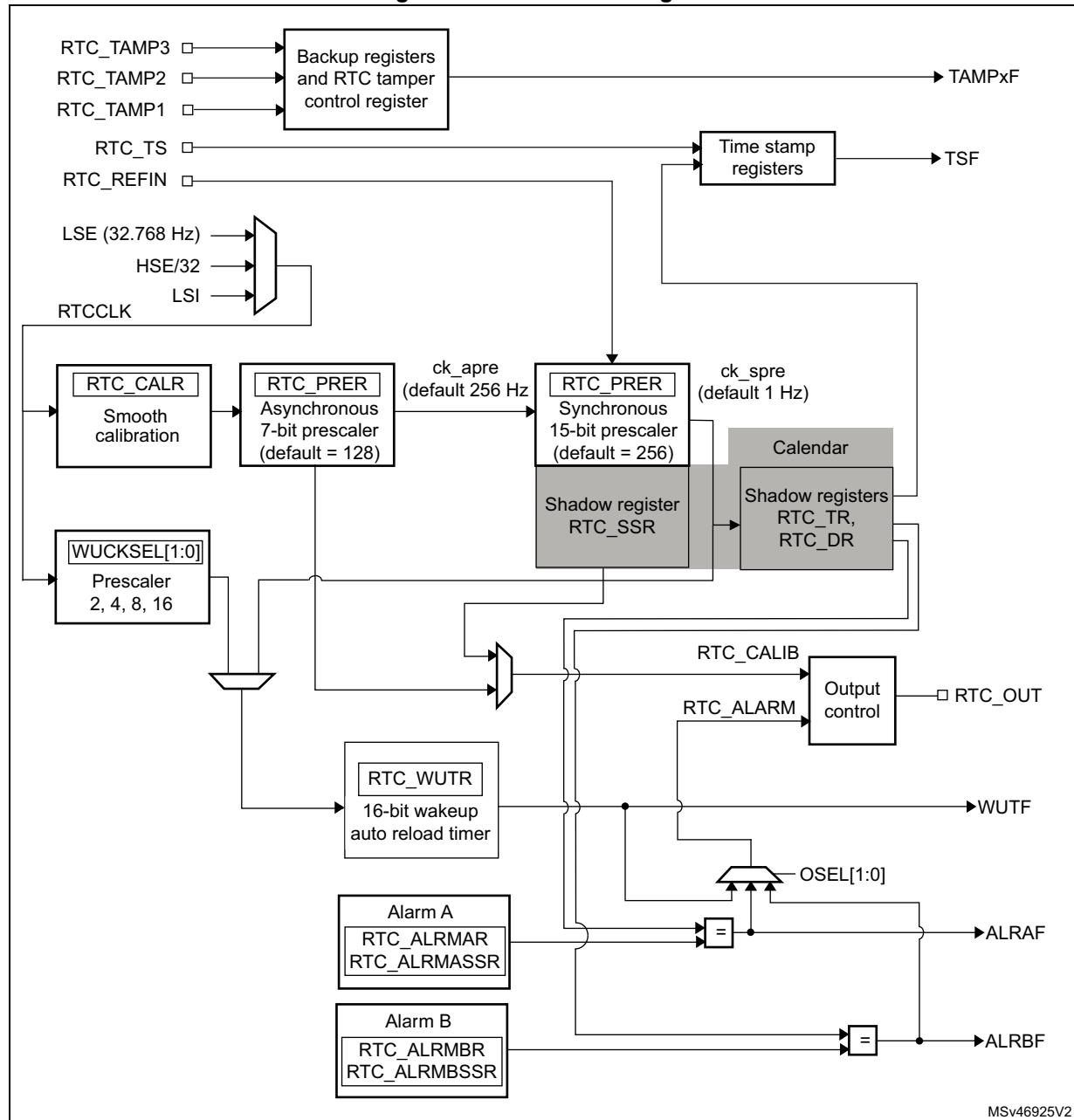
Table 181. RTC implementation

Feature	STM32WB55xx	STM32WB35xx
RTC_TS pin	X	-
RTC_TAMP1	X	-
RTC_TAMP2	X	X
RTC_TAMP3	X	-
RTC_OUT on PC13	X	-
RTC_OUT on PB2	X	X

## 29.4 RTC functional description

### 29.4.1 RTC block diagram

Figure 286. RTC block diagram



MSv46925V2

The RTC includes:

- Tamper detection erases the backup registers.
- One timestamp event from I/O
- Tamper event detection can generate a timestamp event
- Timestamp can be generated when a switch to  $V_{BAT}$  occurs

### 29.4.2 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 8: 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 286: RTC block diagram](#)):

- A 7-bit asynchronous prescaler configured through the PREDIV\_A bits of the RTC\_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV\_S bits of the RTC\_PRER register.

**Note:** *When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.*

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck\_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is  $2^{22}$ .

This corresponds to a maximum input frequency of around 4 MHz.

$f_{ck\_apre}$  is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{\text{PREDIV\_A} + 1}$$

The  $ck\_apre$  clock is used to clock the binary RTC\_SSR subseconds downcounter. When it reaches 0, RTC\_SSR is reloaded with the content of PREDIV\_S.

$f_{ck\_spre}$  is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(\text{PREDIV\_S} + 1) \times (\text{PREDIV\_A} + 1)}$$

The  $ck\_spre$  clock can be used either to update the calendar or as timebase for the 16-bit wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see [Section 29.4.5: Periodic auto-wake-up](#) for details).

### 29.4.3 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC\_SSR for the subseconds
- RTC\_TR for the time
- RTC\_DR for the date

Every RTCCLK period, the current calendar value is copied into the shadow registers, and the RSF bit of RTC\_ISR register is set (see [Section 29.7.4: RTC initialization and status](#)

*register (RTC\_ISR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 1 RTCCLK period.

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.

#### 29.4.4 Programmable alarms

The RTC unit provides programmable alarm: Alarm A and Alarm B. The description below is given for Alarm A, but can be translated in the same way for Alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC\_CR register. The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC\_ALRMASSR and RTC\_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC\_ALRMAR register, and through the MASKSSx bits of the RTC\_ALRMASSR register. The alarm interrupt is enabled through the ALRAIE bit in the RTC\_CR register.

**Caution:** If the seconds field is selected (MSK1 bit reset in RTC\_ALRMAR), the synchronous prescaler division factor set in the RTC\_PRER register must be at least 3 to ensure correct behavior.

Alarm A and Alarm B (if enabled by bits OSEL[1:0] in RTC\_CR register) can be routed to the RTC\_ALARM output. RTC\_ALARM output polarity can be configured through bit POL the RTC\_CR register.

#### 29.4.5 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC\_CR register.

The wake-up timer clock input can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

When RTCCLK is LSE(32.768 kHz), this allows to configure the wake-up interrupt period from 122  $\mu$ s to 32 s, with a resolution down to 61  $\mu$ s.

- ck\_spre (usually 1 Hz internal clock)

When ck\_spre frequency is 1Hz, this allows to achieve a wake-up time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:

- from 1s to 18 hours when WUCKSEL [2:1] = 10
- and from around 18h to 36h when WUCKSEL[2:1] = 11. In this last case 2<sup>16</sup> is added to the 16-bit counter current value. When the initialization sequence is complete (see [Programming the wake-up timer on page 917](#)), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in

the RTC\_ISR register, and the wake-up counter is automatically reloaded with its reload value (RTC\_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC\_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the RTC\_ALARM output provided it has been enabled through bits OSEL[1:0] of RTC\_CR register. RTC\_ALARM output polarity can be configured through the POL bit in the RTC\_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wake-up timer.

## 29.4.6 RTC initialization and configuration

### RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

### RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by clearing the DBP bit in the PWR\_CR register (refer to the power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, all the RTC registers are write-protected. Writing to the RTC registers is enabled by writing a key into the Write Protection register, RTC\_WPR.

The following steps are required to unlock the write protection on all the RTC registers except for RTC\_TAMPxR, RTC\_BKPxR, RTC\_OR and RTC\_ISR[13:8].

1. Write '0xCA' into the RTC\_WPR register.
2. Write '0x53' into the RTC\_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

### Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC\_ISR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC\_ISR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC\_PRER register.
4. Load the initial time and date values in the shadow registers (RTC\_TR and RTC\_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC\_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

**Note:**

After a system reset, the application can read the INITS flag in the RTC\_ISR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC\_ISR register.

### Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC\_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

### Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for Alarm A but can be translated in the same way for Alarm B.

1. Clear ALRAE in RTC\_CR to disable Alarm A.
2. Program the Alarm A registers (RTC\_ALRMASSR/RTC\_ALRMAR).
3. Set ALRAE in the RTC\_CR register to enable Alarm A again.

**Note:**

Each change of the RTC\_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

### Programming the wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC\_WUTR):

1. Clear WUTE in RTC\_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC\_ISR to make sure the access to wake-up 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 wake-up auto-reload value WUT[15:0], and the wake-up clock selection (WUCKSEL[2:0] bits in RTC\_CR). Set WUTE in RTC\_CR to enable the timer again. The wake-up timer restarts down-counting. The WUTWF bit is cleared up to 2 RTCCLK clock cycles after WUTE is cleared, due to clock synchronization.

## 29.4.7 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 APB clock frequency ( $f_{PCLK}$ ) must be equal to or greater than seven times the RTC clock frequency ( $f_{RTCCLK}$ ). This ensures a secure behavior of the synchronization mechanism.

If the APB 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 APB clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC\_ISR register each time the calendar registers are copied into the RTC\_SSR, RTC\_TR and RTC\_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the 3 values, reading either RTC\_SSR or RTC\_TR locks the values in the higher-order calendar shadow registers until RTC\_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK period: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC\_SSR, RTC\_TR and RTC\_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to [Calendar initialization and configuration on page 916](#)): the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

After synchronization (refer to [Section 29.4.9: 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.*

## 29.4.8 Resetting the RTC

The calendar shadow registers (RTC\_SSR, RTC\_TR and RTC\_DR) and some bits of the RTC status register (RTC\_ISR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC\_CR), the prescaler register (RTC\_PRER), the RTC calibration register (RTC\_CALR), the RTC shift register (RTC\_SHIFTR), the RTC timestamp registers

(RTC\_TSSSR, RTC\_TSTR and RTC\_TSDFR), the RTC tamper configuration register (RTC\_TAMPCCR), the RTC backup registers (RTC\_BKPxR), the wake-up timer register (RTC\_WUTR), the Alarm A and Alarm B registers (RTC\_ALRMASSR/RTC\_ALRMAR and RTC\_ALRMBSSR/RTC\_ALRMBR), and the Option register (RTC\_OR).

In addition, when it is clocked by the LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to the RTC clock section of the Reset and clock controller for details on the list of RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

#### 29.4.9 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC\_SSR or RTC\_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC\_SHIFTR.

RTC\_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of  $1 / (\text{PREDIV\_S} + 1)$  seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV\_S[14:0]. The maximum resolution allowed (30.52  $\mu$ s with a 32768 Hz clock) is obtained with PREDIV\_S set to 0x7FFF.

However, increasing PREDIV\_S means that PREDIV\_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC\_SHIFTR). Writing to RTC\_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of  $1 / (\text{PREDIV\_S} + 1)$  seconds. The shift operation consists of adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this will delay the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this will advance the clock.

**Caution:** Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow will occur.

As soon as a shift operation is initiated by a write to the RTC\_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

**Caution:** This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC\_SHIFTR when REFCKON=1.

#### 29.4.10 RTC reference clock detection

The update of the RTC calendar can be synchronized to a reference clock, RTC\_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC\_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC\_REFIN detection is enabled (REFCKON bit of RTC\_CR set to 1), the calendar is still clocked by the LSE, and RTC\_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC\_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck\_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck\_apre periods when detecting the first reference clock edge. A smaller window of 3 ck\_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the synchronous prescaler which outputs the ck\_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck\_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck\_apre period detection window centered on the ck\_spre edge.

When the RTC\_REFIN detection is enabled, PREDIV\_A and PREDIV\_S must be set to their default values:

- PREDIV\_A = 0x007F
- PREDIV\_S = 0x00FF

*Note:* *RTC\_REFIN clock detection is not available in Standby mode.*

#### 29.4.11 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about  $2^{20}$  RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, cal\_cnt[19:0], clocked by RTCCLK.

The smooth calibration register (RTC\_CALR) specifies the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

*Note:* *CALM[8:0] (RTC\_CALR) specifies the number of RTCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to '1' causes exactly one pulse to be masked during the 32-second cycle at the moment when cal\_cnt[19:0] is 0x80000; CALM[1]=1 causes two other cycles to be masked (when cal\_cnt is 0x40000 and 0xC0000); CALM[2]=1*

causes four other cycles to be masked ( $cal\_cnt = 0x20000/0x60000/0xA0000/0xE0000$ ); and so on up to CALM[8]=1 which causes 256 clocks to be masked ( $cal\_cnt = 0xXX800$ ).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to '1' effectively inserts an extra RTCCLK pulse every  $2^{11}$  RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency ( $F_{CAL}$ ) given the input frequency ( $F_{RTCCLK}$ ) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

### Calibration when PREDIV\_A<3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV\_A bits in RTC\_PRER register) is less than 3. If CALP was already set to 1 and PREDIV\_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV\_A less than 3, the synchronous prescaler value (PREDIV\_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV\_A equals 1 (division factor of 2), PREDIV\_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV\_A equals 0, PREDIV\_S should be set to 32759 rather than 32767 (8 less).

If PREDIV\_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

### Verifying the RTC calibration

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC\_CALR register can be set to 1 to force a 16- second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC\_CALR register can be set to 1 to force a 8- second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

### Re-calibration on-the-fly

The calibration register (RTC\_CALR) can be updated on-the-fly while RTC\_ISR/INITF=0, by using the follow process:

1. Poll the RTC\_ISR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC\_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck\_apre cycles after the write operation to RTC\_CALR, the new calibration settings take effect.

## 29.4.12 Time-stamp function

Time-stamp is enabled by setting the TSE or ITSE bits of RTC\_CR register to 1.

When TSE is set:

The calendar is saved in the time-stamp registers (RTC\_TSSSR, RTC\_TSTR, RTC\_TSDR) when a time-stamp event is detected on the RTC\_TS pin.

When ITSE is set:

The calendar is saved in the time-stamp registers (RTC\_TSSSR, RTC\_TSTR, RTC\_TSDR) when an internal time-stamp event is detected. The internal timestamp event is generated by the switch to the VBAT supply.

When a time-stamp event occurs, due to internal or external event, the time-stamp flag bit (TSF) in RTC\_ISR register is set. In case the event is internal, the ITSF flag is also set in RTC\_ISR register.

By setting the TSIE bit in the RTC\_CR register, an interrupt is generated when a time-stamp event occurs.

If a new time-stamp event is detected while the time-stamp flag (TSF) is already set, the time-stamp overflow flag (TSOVF) flag is set and the time-stamp registers (RTC\_TSTR and RTC\_TSDR) maintain the results of the previous event.

**Note:** *TSF is set 2 ck\_apre cycles after the time-stamp event occurs due to synchronization process.*

*There is no delay in the setting of TSOVF. This means that if two time-stamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.*

**Caution:** If a time-stamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a time-stamp event occurring at the same moment, the application must not write '0' into TSF bit unless it has already read it to '1'.

Optionally, a tamper event can cause a time-stamp to be recorded. See the description of the TAMPTS control bit in [Section 29.7.16: RTC tamper configuration register \(RTC\\_TAMPSCR\)](#).

### 29.4.13 Tamper detection

The RTC\_TAMPx input events can be configured either for edge detection, or for level detection with filtering.

The tamper detection can be configured for the following purposes:

- erase the RTC backup registers (default configuration)
- generate an interrupt, capable to wake-up from Stop and Standby modes
- generate a hardware trigger for the low-power timers

#### RTC backup registers

The backup registers (RTC\_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs (see [Section 29.7.20: RTC backup registers \(RTC\\_BKPxR\)](#) and [Tamper detection initialization on page 923](#)) except if the TAMPxNOERASE bit is set, or if TAMPxMF is set in the RTC\_TAMPSCR register.

#### Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the RTC\_TAMPSCR register.

Each RTC\_TAMPx tamper detection input is associated with a flag TAMPxF in the RTC\_ISR register.

When TAMPxMF is cleared:

The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:

- 3 ck\_apre cycles when TAMPFLT differs from 0x0 (Level detection with filtering)
- 3 ck\_apre cycles when TAMPTS=1 (Timestamp on tamper event)
- No latency when TAMPFLT=0x0 (Edge detection) and TAMPTS=0

A new tamper occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

When TAMPxMF is set:

A new tamper occurring on the same pin cannot be detected during the latency described above and 2.5 ck\_rtc additional cycles.

By setting the TAMPIE bit in the RTC\_TAMPCCR register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPIE is not allowed when one or more TAMPxMF is set.

When TAMPIE is cleared, each tamper pin event interrupt can be individually enabled by setting the corresponding TAMPxIE bit in the RTC\_TAMPCCR register. Setting TAMPxIE is not allowed when the corresponding TAMPxMF is set.

### Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMF bit is cleared in RTC\_TAMPCCR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMF bit is set, the TAMPxF flag is masked, and kept cleared in RTC\_ISR register. This configuration allows to trig automatically the low-power timers in Stop mode, without requiring the system wakeup to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

### Timestamp on tamper event

With TAMPTS set to '1', any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit are set in RTC\_ISR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set at the same time that TSF or TSOVF is set.

### Edge detection on tamper inputs

If the TAMPFLT bits are "00", the RTC\_TAMPx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the RTC\_TAMPx inputs are deactivated when edge detection is selected.

**Caution:** When using the edge detection, it is recommended to check by software the tamper pin level just after enabling the tamper detection (by reading the GPIO registers), and before writing sensitive values in the backup registers, to ensure that an active edge did not occur before enabling the tamper event detection.

When TAMPFLT="00" and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the RTC\_TAMPx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (RTC\_BKPxR). This prevents the application from writing to the backup registers while the RTC\_TAMPx input value still indicates a tamper detection. This is equivalent to a level detection on the RTC\_TAMPx input.

**Note:** *Tamper detection is still active when V<sub>DD</sub> power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the RTC\_TAMPx is mapped should be externally tied to the correct level.*

### Level detection with filtering on RTC\_TAMPx inputs

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The RTC\_TAMPx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the RTC\_TAMPx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

*Note:* Refer to the datasheets for the electrical characteristics of the pull-up resistors.

#### 29.4.14 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 duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV\_S+1” is a non-zero multiple of 256 (i.e: PREDIV\_S[7:0] = 0xFF), the RTC\_CALIB frequency is  $f_{RTCCLK}/(256 * (PREDIV_A+1))$ . 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. The 1 Hz output is affected when a shift operation is on going and may toggle during the shift operation (SHPF=1).

*Note:* When COSEL bit is cleared, the RTC\_CALIB output is the output of the 6th stage of the asynchronous prescaler.

*When COSEL bit is set, the RTC\_CALIB output is the output of the 8th stage of the synchronous prescaler.*

#### 29.4.15 Alarm output

The OSEL[1:0] control bits in the RTC\_CR register are used to activate the alarm output RTC\_ALARM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC\_ISR register.

The polarity of the output is determined by the POL control bit in RTC\_CR so that the opposite of the selected flag bit is output when POL is set to 1.

##### Alarm output

The RTC\_ALARM pin can be configured in output open drain or output push-pull using RTC\_OR register.

*Note:* Once the RTC\_ALARM output is enabled, it has priority over RTC\_CALIB (COE bit is don't care and must be kept cleared).

## 29.5 RTC low-power modes

Table 182. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Low-power run	No effect.
Low-power sleep	No effect. RTC interrupts cause the device to exit the Low-power sleep mode.
Stop 0	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Stop mode.
Stop 1	
Stop 2	
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Standby mode.
Shutdown	The RTC remains active when the RTC clock source is LSE. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Shutdown mode.

## 29.6 RTC interrupts

All RTC interrupts are connected to the EXTI controller. Refer to [Section 14: Extended interrupt and event controller \(EXTI\)](#).

Table 183. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit from Sleep mode	Exit from Stop mode	Exit from Standby mode
Alarm A	ALRAF	ALRAIE	Yes	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>
Alarm B	ALRBF	ALRBIE	Yes	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>
RTC_TS input (timestamp)	TSF	TSIE	Yes	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>
RTC_TAMP1 input detection	TAMP1F	TAMPIE	Yes	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>
RTC_TAMP2 input detection	TAMP2F	TAMPIE	Yes	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>
RTC_TAMP3 input detection	TAMP3F	TAMPIE	Yes	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>
Wakeup timer interrupt	WUTF	WUTIE	Yes	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>

1. Wakeup from STOP and Standby modes is possible only when the RTC clock source is LSE or LSI.

## 29.7 RTC registers

Refer to [Section 1.2 on page 61](#) of the reference manual for a list of abbreviations used in register descriptions.

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

### 29.7.1 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 916](#) and [Reading the calendar on page 917](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 916](#).

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

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

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

Bit 22 **PM**: AM/PM notation

- 0: AM or 24-hour format
- 1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

### 29.7.2 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 916](#) and [Reading the calendar on page 917](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 916](#).

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset: 0x0000 2101 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units

000: forbidden

001: Monday

...

111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

### 29.7.3 RTC control register (RTC\_CR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
							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	Res.	FMT	BYPS HAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

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

Bit 24 **ITSE**: timestamp on internal event enable

0: internal event timestamp disabled  
1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable

This bit enables the RTC\_CALIB output  
0: Calibration output disabled  
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to RTC\_ALARM output

00: Output disabled  
01: Alarm A output enabled  
10: Alarm B output enabled  
11: Wake-up output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of RTC\_ALARM output

0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0])  
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]).

Bit 19 **COSEL**: Calibration output selection

When COE=1, this bit selects which signal is output on RTC\_CALIB.

0: Calibration output is 512 Hz (with default prescaler setting)  
1: Calibration output is 1 Hz (with default prescaler setting)

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 29.4.14: 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, 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 outside initialization mode.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set, 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 outside initialization mode.

Bit 15 **TSIE**: Time-stamp interrupt enable

0: Time-stamp Interrupt disable

1: Time-stamp Interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable

0: Wake-up timer interrupt disabled

1: Wake-up timer interrupt enabled

Bit 13 **ALRBIE**: *Alarm B interrupt enable*

0: Alarm B Interrupt disable

1: Alarm B Interrupt enable

Bit 12 **ALRAIE**: *Alarm A interrupt enable*

0: Alarm A interrupt disabled

1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable

0: timestamp disable

1: timestamp enable

Bit 10 **WUTE**: Wake-up timer enable

0: Wake-up timer disabled

1: Wake-up timer enabled

*Note: When the wake-up timer is disabled, wait for WUTWF=1 before enabling it again.*

Bit 9 **ALRBE**: *Alarm B enable*

0: Alarm B disabled

1: Alarm B enabled

Bit 8 **ALRAE**: *Alarm A enable*

0: Alarm A disabled

1: Alarm A enabled

## Bit 7 Reserved, must be kept at reset value.

Bit 6 **FMT**: Hour format

0: 24 hour/day format

1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC\_SSR, RTC\_TR, and RTC\_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC\_SSR, RTC\_TR, and RTC\_DR) are taken directly from the calendar counters.

*Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to '1'.*

Bit 4 **REFCKON**: RTC\_REFIN reference clock detection enable (50 or 60 Hz)

0: RTC\_REFIN detection disabled

1: RTC\_REFIN detection enabled

*Note: PREDIV\_S must be 0x00FF.*

Bit 3 **TSEDGE**: Time-stamp event active edge

0: RTC\_TS input rising edge generates a time-stamp event

1: RTC\_TS input falling edge generates a time-stamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: Wake-up clock selection

000: RTC/16 clock is selected

001: RTC/8 clock is selected

010: RTC/4 clock is selected

011: RTC/2 clock is selected

10x: ck\_spre (usually 1 Hz) clock is selected

11x: ck\_spre (usually 1 Hz) clock is selected and  $2^{16}$  is added to the WUT counter value (see note below)

**Note:** Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC\_ISR/INITF = 1).

WUT = Wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC\_CR WUTE bit = 0 and RTC\_ISR WUTWF bit = 1.

*It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.*

*ADD1H and SUB1H changes are effective in the next second.*

*This register is write protected. The write access procedure is described in [RTC register write protection on page 916](#).*

**Caution:** TSE must be reset when TSEDGE is changed to avoid spuriously setting of TSF.

### 29.7.4 RTC initialization and status register (RTC\_ISR)

This register is write protected (except for RTC\_ISR[13:8] bits). The write access procedure is described in [RTC register write protection on page 916](#).

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to '0'

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSF	RECALPF
														rc_w0	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRB WF	ALRAWF
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **ITSF**: Internal tTime-stamp flag

This flag is set by hardware when a time-stamp on the internal event occurs.

This flag is cleared by software by writing 0, and must be cleared together with TSF bit by writing 0 in both bits.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to '1' when software writes to the RTC\_CALR register, indicating that the RTC\_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to '0'. Refer to [Re-calibration on-the-fly](#).

Bit 15 **TAMP3F**: RTC\_TAMP3 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC\_TAMP3 input.

It is cleared by software writing 0

Bit 14 **TAMP2F**: RTC\_TAMP2 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC\_TAMP2 input.

It is cleared by software writing 0

Bit 13 **TAMP1F**: RTC\_TAMP1 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC\_TAMP1 input.

It is cleared by software writing 0

Bit 12 **TSOVF**: Time-stamp overflow flag

This flag is set by hardware when a time-stamp event occurs while TSF is already set.

This flag is cleared by software by writing 0. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a time-stamp event occurs immediately before the TSF bit is cleared.

Bit 11 **TSF**: Time-stamp flag

This flag is set by hardware when a time-stamp event occurs.

This flag is cleared by software by writing 0. If ITSF flag is set, TSF must be cleared together with ITSF by writing 0 in both bits.

**Bit 10 WUTF:** Wake-up timer flag

This flag is set by hardware when the wake-up 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\_SSR, RTC\_TR and RTC\_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow register mode (BYPSHAD=1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

**Bit 4 INITS:** Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

**Bit 3 SHPF:** Shift operation pending

0: No shift operation is pending

1: A shift operation is pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC\_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

**Bit 2 WUTWF:** Wake-up timer write flag

This bit is set by hardware up to 2 RTCCLK cycles after the WUTE bit has been set to 0 in RTC\_CR, and is cleared up to 2 RTCCLK cycles after the WUTE bit has been set to 1. The wake-up timer values can be changed when WUTE bit is cleared and WUTWF is set.

- 0: Wake-up timer configuration update not allowed
- 1: Wake-up timer configuration update allowed

**Bit 1 ALRBWF:** Alarm B write flag

This bit is set by hardware when Alarm B values can be changed, after the ALRBE bit has been set to 0 in RTC\_CR.

It is cleared by hardware in initialization mode.

- 0: Alarm B update not allowed
- 1: Alarm B update allowed

**Bit 0 ALRAWF:** Alarm A write flag

This bit is set by hardware when Alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC\_CR.

It is cleared by hardware in initialization mode.

- 0: Alarm A update not allowed
- 1: Alarm A update allowed

**Note:** *The bits ALRAF, ALRBF, WUTF and TSF are cleared 2 APB clock cycles after programming them to 0.*

### 29.7.5 RTC prescaler register (RTC\_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to [Calendar initialization and configuration on page 916](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 916](#).

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

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

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

Bits 22:16 **PREDIV\_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:

$ck_{apre}$  frequency = RTCCLK frequency/(PREDIV\_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV\_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:

$ck_{spre}$  frequency =  $ck_{apre}$  frequency/(PREDIV\_S+1)

## 29.7.6 RTC wake-up timer register (RTC\_WUTR)

This register can be written only when WUTWF is set to 1 in RTC\_ISR.

This register is write protected. The write access procedure is described in [RTC register write protection on page 916](#).

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

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

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

Bits 15:0 **WUT[15:0]**: Wake-up auto-reload value bits

When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck\_wut cycles. The ck\_wut period is selected through WUCKSEL[2:0] bits of the RTC\_CR register

When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs (WUT+1) ck\_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] =011 (RTCCLK/2) is forbidden.

### 29.7.7 RTC alarm A register (RTC\_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC\_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 916](#).

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask

- 0: Alarm A set if the date/day match
- 1: Date/day don't care in Alarm A comparison

Bit 30 **WDSEL**: Week day selection

- 0: DU[3:0] represents the date units
- 1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format.

Bits 27:24 **DU[3:0]**: Date units or day in BCD format.

Bit 23 **MSK3**: Alarm A hours mask

- 0: Alarm A set if the hours match
- 1: Hours don't care in Alarm A comparison

Bit 22 **PM**: AM/PM notation

- 0: AM or 24-hour format
- 1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 **MSK2**: Alarm A minutes mask

- 0: Alarm A set if the minutes match
- 1: Minutes don't care in Alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 **MSK1**: Alarm A seconds mask

- 0: Alarm A set if the seconds match
- 1: Seconds don't care in Alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

### 29.7.8 RTC alarm B register (RTC\_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC\_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 916](#).

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]		SU[3:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask

- 0: Alarm B set if the date and day match
- 1: Date and day don't care in Alarm B comparison

Bit 30 **WDSEL**: Week day selection

- 0: DU[3:0] represents the date units
- 1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask

- 0: Alarm B set if the hours match
- 1: Hours don't care in Alarm B comparison

Bit 22 **PM**: AM/PM notation

- 0: AM or 24-hour format
- 1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask

- 0: Alarm B set if the minutes match
- 1: Minutes don't care in Alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask

- 0: Alarm B set if the seconds match
- 1: Seconds don't care in Alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

### 29.7.9 RTC write protection register (RTC\_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to [RTC register write protection](#) for a description of how to unlock RTC register write protection.

### 29.7.10 RTC sub second register (RTC\_SSR)

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

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

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

Bits 15:0 **SS[15:0]**: Sub second value

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV\_S - SS) / (PREDIV\_S + 1)

*Note: SS can be larger than PREDIV\_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC\_TR/RTC\_DR.*

### 29.7.11 RTC shift control register (RTC\_SHIFTR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 916](#).

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC\_ISR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC\_ISR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV\_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV\_S + 1))).

*Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF=1 to be sure that the shadow registers have been updated with the shifted time.*

### 29.7.12 RTC timestamp time register (RTC\_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC\_ISR. It is cleared when TSF bit is reset.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

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

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

### 29.7.13 RTC timestamp date register (RTC\_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC\_ISR. It is cleared when TSF bit is reset.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

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

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

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

### 29.7.14 RTC time-stamp sub second register (RTC\_TSSSR)

The content of this register is valid only when RTC\_ISR/TSF is set. It is cleared when the RTC\_ISR/TSF bit is reset.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **SS[15:0]**: Sub second value

SS[15:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

### 29.7.15 RTC calibration register (RTC\_CALR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 916](#).

Address offset: 0x3C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every  $2^{11}$  pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. If the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:  $(512 * \text{CALP}) - \text{CALM}$ .

Refer to [Section 29.4.11: 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.

*Note:* CALM[1:0] are stuck at "00" when CALW8='1'. Refer to [Section 29.4.11: 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 stuck at '0' when CALW16='1'. Refer to [Section 29.4.11: RTC smooth digital calibration](#).

Bits 12:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of  $2^{20}$  RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See [Section 29.4.11: RTC smooth digital calibration on page 920](#).

### 29.7.16 RTC tamper configuration register (RTC\_TAMPCCR)

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3 MF	TAMP3 NO ERASE	TAMP3 IE	TAMP2 MF	TAMP2 NO ERASE	TAMP2 IE	TAMP1 MF	TAMP1 NO ERASE	TAMP1 IE
							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
TAMP PUDIS	TAMP PRCH [1:0]	TAMP FLT[1:0]	TAMP FREQ[2:0]	TAMP TS	TAMP3 TRG	TAMP3 E	TAMP2 TRG	TAMP2 E	TAMP1 E	TAMP1 TRG	TAMP1 E				
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.

Bit 24 **TAMP3MF**: Tamper 3 mask flag

- 0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.
- 1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers are not erased.

*Note: The Tamper 3 interrupt must not be enabled when TAMP3MF is set.*

Bit 23 **TAMP3NOERASE**: Tamper 3 no erase

- 0: Tamper 3 event erases the backup registers.
- 1: Tamper 3 event does not erase the backup registers.

Bit 22 **TAMP3IE**: Tamper 3 interrupt enable

- 0: Tamper 3 interrupt is disabled if TAMP1IE = 0.
- 1: Tamper 3 interrupt enabled.

Bit 21 **TAMP2MF**: Tamper 2 mask flag

- 0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.
- 1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers are not erased.

*Note: The Tamper 2 interrupt must not be enabled when TAMP2MF is set.*

Bit 20 **TAMP2NOERASE**: Tamper 2 no erase

- 0: Tamper 2 event erases the backup registers.
- 1: Tamper 2 event does not erase the backup registers.

Bit 19 **TAMP2IE**: Tamper 2 interrupt enable

- 0: Tamper 2 interrupt is disabled if TAMP1IE = 0.
- 1: Tamper 2 interrupt enabled.

Bit 18 **TAMP1MF**: Tamper 1 mask flag

- 0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.
- 1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers are not erased.

*Note: The Tamper 1 interrupt must not be enabled when TAMP1MF is set.*

- Bit 17 **TAMP1NOERASE**: Tamper 1 no erase  
 0: Tamper 1 event erases the backup registers.  
 1: Tamper 1 event does not erase the backup registers.
- Bit 16 **TAMP1IE**: Tamper 1 interrupt enable  
 0: Tamper 1 interrupt is disabled if TAMP1IE = 0.  
 1: Tamper 1 interrupt enabled.
- Bit 15 **TAMPPUDIS**: RTC\_TAMPx pull-up disable  
 This bit determines if each of the RTC\_TAMPx pins are precharged before each sample.  
 0: Precharge RTC\_TAMPx pins before sampling (enable internal pull-up)  
 1: Disable precharge of RTC\_TAMPx pins.
- Bits 14:13 **TAMPPRCH[1:0]**: RTC\_TAMPx precharge duration  
 These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the RTC\_TAMPx inputs.  
 0x0: 1 RTCCLK cycle  
 0x1: 2 RTCCLK cycles  
 0x2: 4 RTCCLK cycles  
 0x3: 8 RTCCLK cycles
- Bits 12:11 **TAMPFLT[1:0]**: RTC\_TAMPx filter count  
 These bits determines the number of consecutive samples at the specified level (TAMP\*TRG) needed to activate a Tamper event. TAMPFLT is valid for each of the RTC\_TAMPx inputs.  
 0x0: Tamper event is activated on edge of RTC\_TAMPx input transitions to the active level (no internal pull-up on RTC\_TAMPx input).  
 0x1: Tamper event is activated after 2 consecutive samples at the active level.  
 0x2: Tamper event is activated after 4 consecutive samples at the active level.  
 0x3: Tamper event is activated after 8 consecutive samples at the active level.
- Bits 10:8 **TAMPFREQ[2:0]**: Tamper sampling frequency  
 Determines the frequency at which each of the RTC\_TAMPx inputs are sampled.  
 0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)  
 0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)  
 0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)  
 0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)  
 0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)  
 0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)  
 0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)  
 0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)
- Bit 7 **TAMPPTS**: 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  
 TAMPPTS is valid even if TSE=0 in the RTC\_CR register.
- Bit 6 **TAMP3TRG**: Active level for RTC\_TAMP3 input  
 if TAMPFLT ≠ 00:  
 0: RTC\_TAMP3 input staying low triggers a tamper detection event.  
 1: RTC\_TAMP3 input staying high triggers a tamper detection event.  
 if TAMPFLT = 00:  
 0: RTC\_TAMP3 input rising edge triggers a tamper detection event.  
 1: RTC\_TAMP3 input falling edge triggers a tamper detection event.

- Bit 5 **TAMP3E**: RTC\_TAMP3 detection enable  
0: RTC\_TAMP3 input detection disabled  
1: RTC\_TAMP3 input detection enabled
- Bit 4 **TAMP2TRG**: Active level for RTC\_TAMP2 input  
if TAMPFLT != 00:  
0: RTC\_TAMP2 input staying low triggers a tamper detection event.  
1: RTC\_TAMP2 input staying high triggers a tamper detection event.  
if TAMPFLT = 00:  
0: RTC\_TAMP2 input rising edge triggers a tamper detection event.  
1: RTC\_TAMP2 input falling edge triggers a tamper detection event.
- Bit 3 **TAMP2E**: RTC\_TAMP2 input detection enable  
0: RTC\_TAMP2 detection disabled  
1: RTC\_TAMP2 detection enabled
- Bit 2 **TAMPIE**: Tamper interrupt enable  
0: Tamper interrupt disabled  
1: Tamper interrupt enabled.  
*Note: This bit enables the interrupt for all tamper pins events, whatever TAMPxIE level. If this bit is cleared, each tamper event interrupt can be individually enabled by setting TAMPxIE.*
- Bit 1 **TAMP1TRG**: Active level for RTC\_TAMP1 input  
If TAMPFLT != 00  
0: RTC\_TAMP1 input staying low triggers a tamper detection event.  
1: RTC\_TAMP1 input staying high triggers a tamper detection event.  
if TAMPFLT = 00:  
0: RTC\_TAMP1 input rising edge triggers a tamper detection event.  
1: RTC\_TAMP1 input falling edge triggers a tamper detection event.
- Bit 0 **TAMP1E**: RTC\_TAMP1 input detection enable  
0: RTC\_TAMP1 detection disabled  
1: RTC\_TAMP1 detection enabled

**Caution:** When TAMPFLT = 0, TAMPxIE must be reset when TAMPxTRG is changed to avoid spuriously setting TAMPxF.

### 29.7.17 RTC alarm A sub second register (RTC\_ALRMASSR)

This register can be written only when ALRAE is reset in RTC\_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 916](#)

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

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

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

Bits 27:24 MASKSS[3:0]: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[14:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[14:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

3: SS[14:3] are don't care in Alarm A comparison. Only SS[2:0] are compared.

...

12: SS[14:12] are don't care in Alarm A comparison. SS[11:0] are compared.

13: SS[14:13] are don't care in Alarm A comparison. SS[12:0] are compared.

14: SS[14] is don't care in Alarm A comparison. SS[13:0] are compared.

15: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 SS[14:0]: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

### 29.7.18 RTC alarm B sub second register (RTC\_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC\_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

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

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

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0x0: No comparison on sub seconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

0x1: SS[14:1] are don't care in Alarm B comparison. Only SS[0] is compared.

0x2: SS[14:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.

0x3: SS[14:3] are don't care in Alarm B comparison. Only SS[2:0] are compared.

...

0xC: SS[14:12] are don't care in Alarm B comparison. SS[11:0] are compared.

0xD: SS[14:13] are don't care in Alarm B comparison. SS[12:0] are compared.

0xE: SS[14] is don't care in Alarm B comparison. SS[13:0] are compared.

0xF: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if Alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

### 29.7.19 RTC option register (RTC\_OR)

Address offset: 0x4C

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RTC\_OUT\_RMP**: RTC\_OUT remap

Setting this bit allows to remap the RTC outputs on PB2 as follows:

**RTC\_OUT\_RMP = '0' (only available on STM32WB55):**

If OSEL/= '00': RTC\_ALARM is output on PC13

If OSEL= '00' and COE = '1': RTC\_CALIB is output on PC13

**RTC\_OUT\_RMP = '1':**

If OSEL /= '00' and COE = '0': RTC\_ALARM is output on PB2

If OSEL = '00' and COE = '1': RTC\_CALIB is output on PB2

If OSEL /= '00' and COE = '1': RTC\_CALIB is output on PB2 and (on STM32WB55 only) RTC\_ALARM is output on PC13.

Bit 0 **RTC\_ALARM\_TYPE**: RTC\_ALARM output type on PC13

This bit is set and cleared by software

0: RTC\_ALARM, when mapped on PC13, is open-drain output

1: RTC\_ALARM, when mapped on PC13, is push-pull output

*Note: This bit is reserved for STM32WB35.*

### 29.7.20 RTC backup registers (RTC\_BKPxR)

Address offset: 0x50 to 0x9C

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 BKP[31:0]

The application can write or read data to and from these registers.

They are powered-on by  $V_{BAT}$  when  $V_{DD}$  is switched off, so that they are not reset by System reset, and their contents remain valid when the device operates in low-power mode. This register is reset on a tamper detection event, as long as  $TAMPxF=1$ .

## 29.7.21 RTC register map

**Table 184. RTC register map and reset values**

Table 184. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x2C	RTC_SHIFTR	ADD1S	Res.																															
	Reset value	0																																
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																																	
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																																	
0x38	RTC_TSRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																																	
0x3C	RTC_CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																																	
0x40	RTC_TAMPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x44	RTC_ALRMASSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x48	RTC_ALRBSSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x4C	RTC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																																	
0x50 to 0x9C	RTC_BKP0R																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	to RTC_BKP19R																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 30 Independent watchdog (IWDG)

### 30.1 Introduction

The devices feature an embedded watchdog peripheral that offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral detects and solves malfunctions due to software failure, and triggers system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to [Section 31 on page 962](#).

### 30.2 IWDG main features

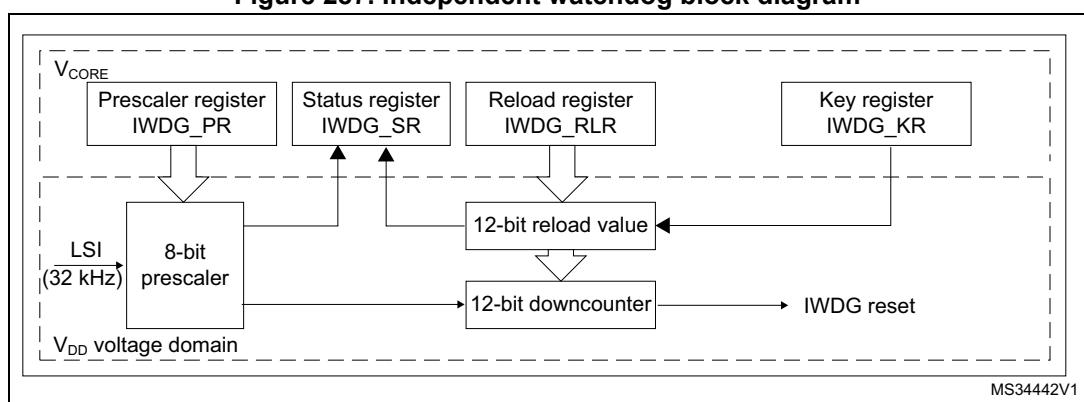
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

### 30.3 IWDG functional description

#### 30.3.1 IWDG block diagram

*Figure 287* shows the functional blocks of the independent watchdog module.

**Figure 287. Independent watchdog block diagram**



1. The register interface is located in the  $V_{CORE}$  voltage domain. The watchdog function is located in the  $V_{DD}$  voltage domain, still functional in Stop and Standby mode.

When the independent watchdog is started by writing the value 0x0000 CCCC in the *IWDG key register (IWDG\_KR)*, the counter starts counting down from the reset value of 0xFFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG\_KR)*, the IWDG\_RLR value is reloaded in the counter and the watchdog reset is prevented.

Once running, the IWDG cannot be stopped.

### 30.3.2 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the *IWDG window register (IWDG\_WINR)*.

If the reload operation is performed while the counter is greater than the value stored in the *IWDG window register (IWDG\_WINR)*, then a reset is provided.

The default value of the *IWDG window register (IWDG\_WINR)* is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the *IWDG reload register (IWDG\_RLR)* value and ease the cycle number calculation to generate the next reload.

#### Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG\_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG\_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG\_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG\_RLR)*.
5. Wait for the registers to be updated (IWDG\_SR = 0x0000 0000).
6. Write to the *IWDG window register (IWDG\_WINR)*. This automatically refreshes the counter value in the *IWDG reload register (IWDG\_RLR)*.

*Note:* Writing the window value allows the counter value to be refreshed by the RLR when *IWDG status register (IWDG\_SR)* is set to 0x0000 0000.

#### Configuring the IWDG when the window option is disabled

When the window option is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG\_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG\_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG\_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG\_RLR)*.
5. Wait for the registers to be updated (IWDG\_SR = 0x0000 0000).
6. Refresh the counter value with IWDG\_RLR (IWDG\_KR = 0x0000 AAAA).

### 30.3.3 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the [IWDG key register \(IWDG\\_KR\)](#) is written by the software before the counter reaches end of count or if the downcounter is reloaded inside the window.

### 30.3.4 Low-power freeze

Depending on the IWDG\_STOP and IWDG\_STBY options configuration, the IWDG can continue counting or not during the Stop mode and the Standby mode, respectively. If the IWDG is kept running during Stop or Standby modes, it can wake up the device from this mode. Refer to [User and read protection option bytes](#) for more details.

### 30.3.5 Register access protection

Write access to [IWDG prescaler register \(IWDG\\_PR\)](#), [IWDG reload register \(IWDG\\_RLR\)](#) and [IWDG window register \(IWDG\\_WINR\)](#) is protected. To modify them, the user must first write the code 0x0000 5555 in the [IWDG key register \(IWDG\\_KR\)](#). A write access to this register with a different value breaks the sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or of the downcounter reload value or of the window value is ongoing.

### 30.3.6 Debug mode

When the CPU1 enters Debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on the configuration of the corresponding bit in DBGMCU freeze register.

## 30.4 IWDG registers

Refer to [Section 1.2 on page 61](#) for a list of abbreviations used in register descriptions.

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

### 30.4.1 IWDG key register (IWDG\_KR)

Address offset: 0x000

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

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

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

Bits 15:0 **KEY[15:0]**: Key value (write only, read 0x0000)

These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 0x5555 to enable access to the IWDG\_PR, IWDG\_RLR and IWDG\_WINR registers (see [Section 30.3.5: Register access protection](#))

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

### 30.4.2 IWDG prescaler register (IWDG\_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PR[2:0]														
															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 30.3.5: Register access protection](#). They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the [IWDG status register \(IWDG\\_SR\)](#) must be reset in order to be able to change the prescaler divider.

- 000: divider /4
- 001: divider /8
- 010: divider /16
- 011: divider /32
- 100: divider /64
- 101: divider /128
- 110: divider /256
- 111: divider /256

*Note: Reading this register returns the prescaler value from the  $V_{DD}$  voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the [IWDG status register \(IWDG\\_SR\)](#) is reset.*

### 30.4.3 IWDG reload register (IWDG\_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

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

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

Bits 11:0 **RL[11:0]**: Watchdog counter reload value

These bits are write access protected see [Register access protection](#). They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the [IWDG key register \(IWDG\\_KR\)](#). The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to the datasheet for the timeout information.

The RVU bit in the [IWDG status register \(IWDG\\_SR\)](#) must be reset to be able to change the reload value.

*Note: Reading this register returns the reload value from the  $V_{DD}$  voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on it. For this reason the value read from this register is valid only when the RVU bit in the [IWDG status register \(IWDG\\_SR\)](#) is reset.*

### 30.4.4 IWDG status register (IWDG\_SR)

Address offset: 0x0C

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

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

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

Bit 2 **WVU**: Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the  $V_{DD}$  voltage domain (takes up to five cycles).

Window value can be updated only when WVU bit is reset.

Bit 1 **RVU**: Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the  $V_{DD}$  voltage domain (takes up to five 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 five cycles).

Prescaler value can be updated only when PVU bit is reset.

Note:

*If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.*

### 30.4.5 IWDG window register (IWDG\_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF (reset by Standby mode)

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

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

Bits 11:0 **WIN[11:0]**: Watchdog counter window value

These bits are write access protected, see [Section 30.3.5](#), they contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0

The WVU bit in the [IWDG status register \(IWDG\\_SR\)](#) must be reset in order to be able to change the reload value.

*Note: Reading this register returns the reload value from the  $V_{DD}$  voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the [IWDG status register \(IWDG\\_SR\)](#) is reset.*

### 30.4.6 IWDG register map

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

**Table 185. IWDG register map and reset values**

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

# 31 System window watchdog (WWDG)

## 31.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter has reached the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications which require the watchdog to react within an accurate timing window.

## 31.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see [Figure 289](#))
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40.

## 31.3 WWDG functional description

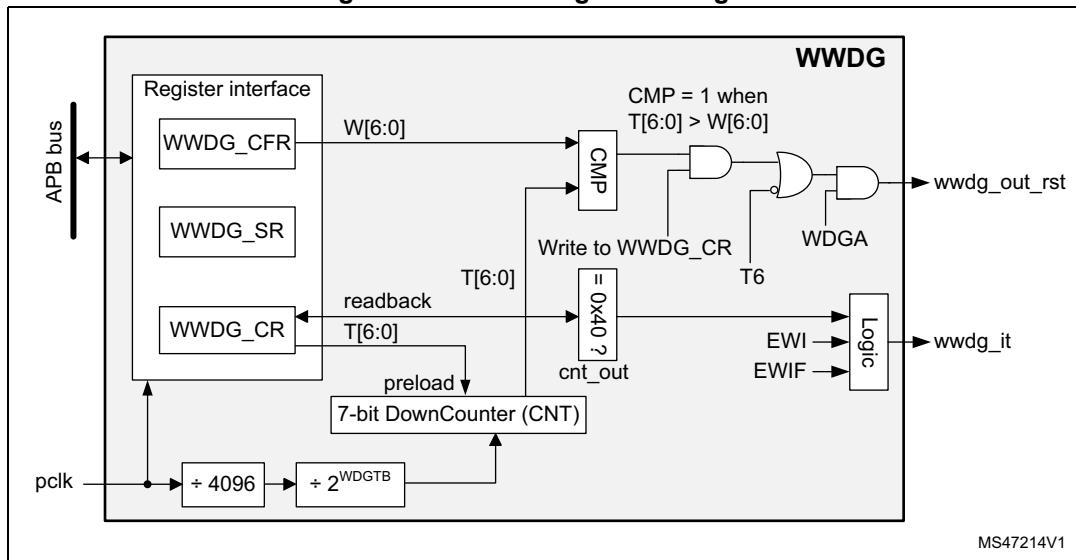
If the watchdog is activated (the WDGA bit is set in the WWDG\_CR register) and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG\_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value and higher than 0x3F. The value to be stored in the WWDG\_CR register must be between 0xFF and 0xC0.

Refer to [Figure 288](#) for the WWDG block diagram.

### 31.3.1 WWDG block diagram

**Figure 288. Watchdog block diagram**



### 31.3.2 Enabling the watchdog

When the user option WWDG\_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG\_CR register, then it cannot be disabled again except by a reset.

When the user option WWDG\_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

### 31.3.3 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG\_CR register (see [Figure 289](#)). The [WWDG configuration register \(WWDG\\_CFR\)](#) contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than the window register value and greater than 0x3F. [Figure 289](#) describes the window watchdog process.

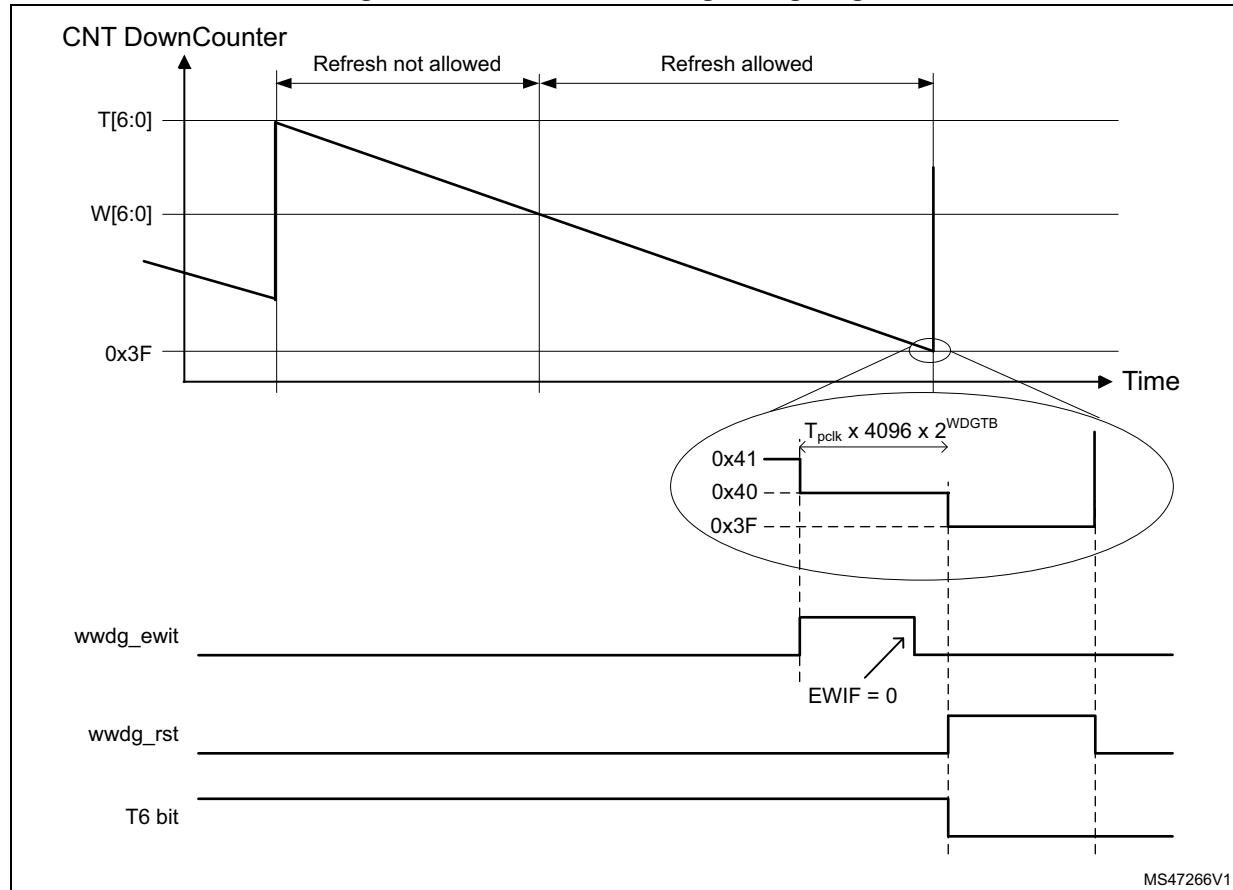
**Note:** The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

### 31.3.4 How to program the watchdog timeout

Use the formula in [Figure 289](#) 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 289. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (\text{ms})$$

where:

$t_{WWDG}$ : WWDG timeout

$t_{PCLK}$ : APB clock period measured in ms

4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3 and T[5:0] is set to 63:

$$t_{WWDG} = (1/48000) \times 4096 \times 2^3 \times (63 + 1) = 43.69\text{ms}$$

Refer to the datasheet for the minimum and maximum values of  $t_{WWDG}$ .

### 31.3.5 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details refer to [Section 41: Debug support \(DBG\)](#).

## 31.4 WWDG interrupts

The early wake-up 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 down-counter 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 ISR has to 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 (for example due to a system lock in a higher priority task) the WWDG reset is eventually generated.*

## 31.5 WWDG registers

Refer to [Section 1.2 on page 61](#) for a list of abbreviations used in register descriptions.

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

### 31.5.1 WWDG control register (WWDG\_CR)

Address offset: 0x000

Reset value: 0x0000 007F

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

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

Bit 7 **WDGA**: Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]**: 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every  $(4096 \times 2^{\text{WDGTB}[2:0]})$  PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

### 31.5.2 WWDG configuration register (WWDG\_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

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

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

Bits 13:11 **WDGTB[2:0]**: Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI**: Early wake-up interrupt

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 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]**: 7-bit window value

These bits contain the window value to be compared with the down-counter.

### 31.5.3 WWDG status register (WWDG\_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EWIF														
															rc_w0

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

Bit 0 **EWIF**: Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

### 31.5.4 WWDG register map

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

Table 186. 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	
0x000	<b>WWDG_CR</b>	Res.																																
	Reset value																																	
0x004	<b>WWDG_CFR</b>	Res.																																
	Reset value																																	
0x008	<b>WWDG_SR</b>	Res.																																
	Reset value																																	

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 32 Inter-integrated circuit (I2C) interface

### 32.1 Introduction

The I<sup>2</sup>C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I<sup>2</sup>C bus. It provides multimaster capability, and controls all I<sup>2</sup>C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus<sup>®</sup> (power management bus) compatible.

DMA can be used to reduce CPU overload.

### 32.2 I2C main features

- I<sup>2</sup>C bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters

The following additional features are also available, depending on the product implementation (see [Section 32.3: I2C implementation](#)):

- SMBus specification rev 3.0 compatibility:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the PCLK reprogramming

- Wake-up from Stop mode on address match.

### 32.3 I2C implementation

Table 187. I2C implementation

I2C features <sup>(1)</sup>	I2C1	I2C3
7-bit addressing mode	X	X
10-bit addressing mode	X	X
Standard mode (up to 100 kbit/s)	X	X
Fast mode (up to 400 kbit/s)	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X
Independent clock	X	X
SMBus/PMbus	X	X
Wakeup from Stop 0 / Stop 1 mode on address match	X	X
Wakeup from Stop 2 mode on address match	-	X

1. X = supported.

### 32.4 I2C functional description

In addition to receiving and transmitting data, this interface converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I<sup>2</sup>C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I<sup>2</sup>C bus.

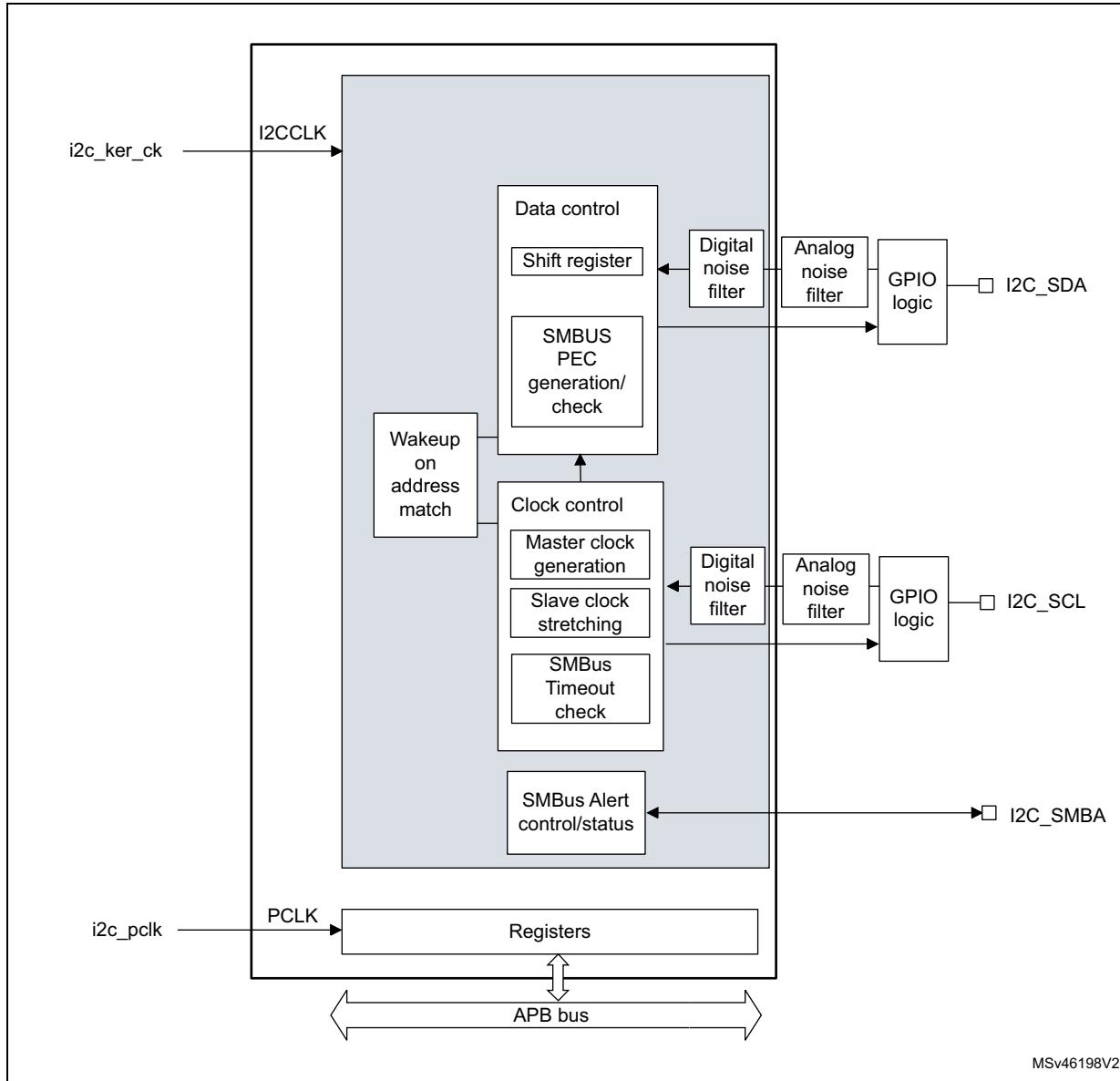
This interface can also be connected to an SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported, the additional optional SMBus Alert pin (SMBA) is also available.

### 32.4.1 I2C block diagram

The block diagram of the I2C interface is shown in [Figure 290](#).

**Figure 290. I2C block diagram**



The I2C is clocked by an independent clock source, which allows the I2C to operate independently from the PCLK frequency.

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration controller (SYSCFG). Refer to [Section 32.3: I2C implementation](#).

### 32.4.2 I2C pins and internal signals

Table 188. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I2C data
I2C_SCL	Bidirectional	I2C clock
I2C_SMBA	Bidirectional	SMBus alert

Table 189. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_it	Output	I2C interrupts, refer to <a href="#">Table 202</a> for the full list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)

### 32.4.3 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period  $t_{I2CCLK}$  must respect the following conditions:

$$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4 \text{ and } t_{I2CCLK} < t_{HIGH}$$

with:

$t_{LOW}$ : SCL low time and  $t_{HIGH}$ : SCL high time

$t_{filters}$ : when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is DNF x  $t_{I2CCLK}$ .

The PCLK clock period  $t_{PCLK}$  must respect the following condition:

$$t_{PCLK} < 4 / 3 t_{SCL}$$

with  $t_{SCL}$ : SCL period

**Caution:** When the I2C kernel is clocked by PCLK, this clock must respect the conditions for  $t_{I2CCLK}$ .

### 32.4.4 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

### Communication flow

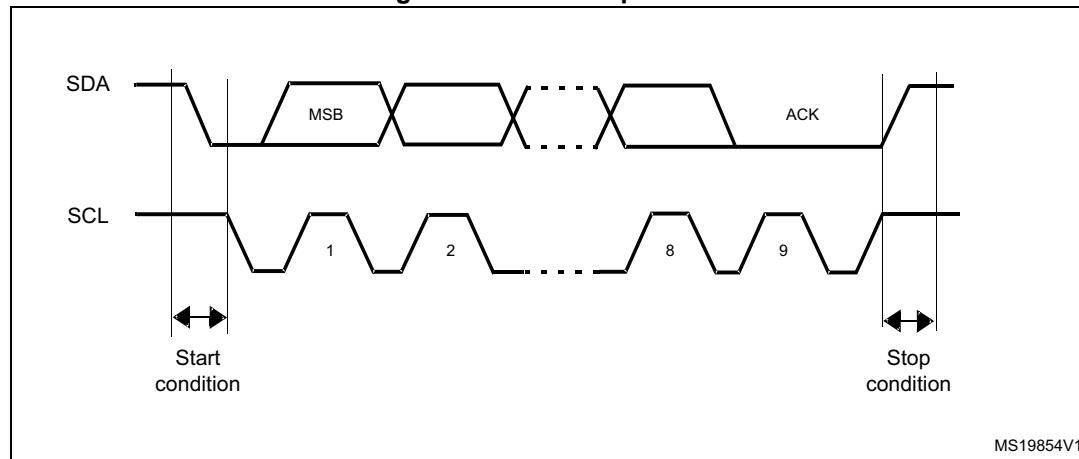
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7- or 10-bit), and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A ninth clock pulse follows the eight clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter (see [Figure 291](#)).

**Figure 291. I<sup>2</sup>C bus protocol**



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

### 32.4.5 I2C initialization

#### Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller, then the I2C can be enabled by setting the PE bit in the I2C\_CR1 register.

When the I2C is disabled (PE = 0), the I<sup>2</sup>C performs a software reset. Refer to [Section 32.4.6](#) for more details.

#### Noise filters

Before enabling the I2C peripheral by setting the PE bit in I2C\_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I<sup>2</sup>C specification, which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The

user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C\_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2CCLK periods. This allows spikes with a programmable length of 1 to 15 I2CCLK periods to be suppressed.

**Table 190. Comparison of analog vs. digital filters**

-	Analog filter	Digital filter
Pulse width of suppressed spikes	$\geq 50$ ns	Programmable length from 1 to 15 I2C peripheral clocks
Benefits	Available in Stop mode	<ul style="list-style-type: none"> <li>– Programmable length: extra filtering capability versus standard requirements</li> <li>– Stable length</li> </ul>
Drawbacks	Variation vs. temperature, voltage, process	Wake-up from Stop mode on address match is not available when digital filter is enabled

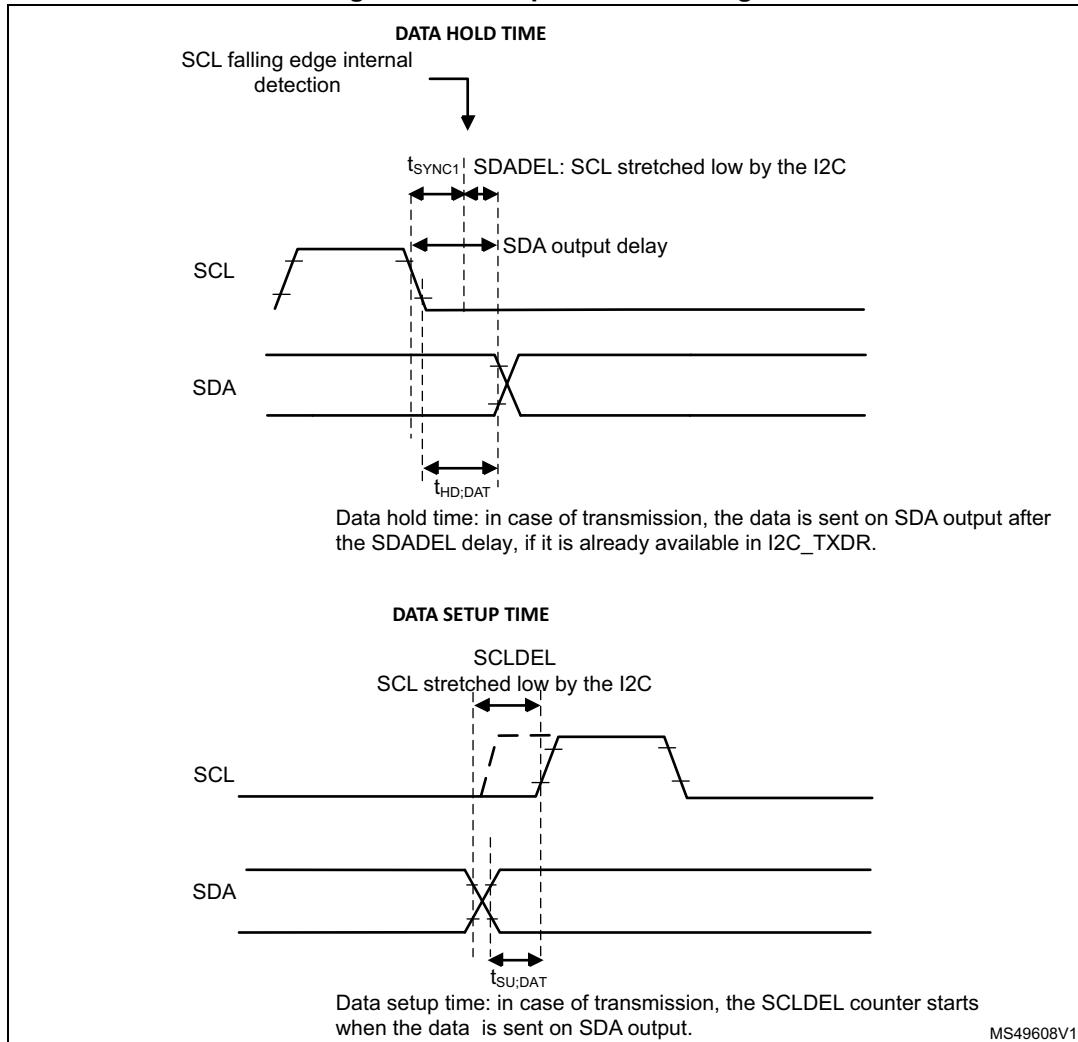
**Caution:** Changing the filter configuration is not allowed when the I2C is enabled.

## I2C timings

The timings must be configured in order to guarantee correct data hold and setup times, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C\_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C\_TIMINGR content in the I2C configuration window.

**Figure 292. Setup and hold timings**



- When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is  $t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK}$  where  $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$ .  
 $t_{SDADEL}$  impacts the hold time  $t_{HD;DAT}$ .

The total SDA output delay is:

$$t_{SYNC1} + \{[SDADEL \times (PRESC+1) + 1] \times t_{I2CCLK}\}$$

$t_{SYNC1}$  duration depends on these parameters:

- SCL falling slope
- When enabled, input delay brought by the analog filter:  $t_{AF(min)} < t_{AF} < t_{AF(max)}$
- When enabled, input delay brought by the digital filter:  $t_{DNF} = DNF \times t_{I2CCLK}$
- Delay due to SCL synchronization to I2CCLK clock (two to three I2CCLK periods)

In order to bridge the undefined region of the SCL falling edge, the user must program SDADEL in such a way that:

$$\{t_f(max) + t_{HD;DAT}(min) - t_{AF(min)} - [(DNF+3) \times t_{I2CCLK}]\} / \{(PRESC+1) \times t_{I2CCLK}\} \leq SDADEL$$

$$SDADEL \leq \{t_{HD;DAT}(max) - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC+1) \times t_{I2CCLK}\}$$

*Note:*  $t_{AF(min)}$  /  $t_{AF(max)}$  are part of the equation only when the analog filter is enabled. Refer to device datasheet for  $t_{AF}$  values.

The maximum  $t_{HD;DAT}$  can be 3.45  $\mu$ s, 0.9  $\mu$ s and 0.45  $\mu$ s for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of  $t_{VD;DAT}$  by a transition time. This maximum must only be met if the device does not stretch the LOW period ( $t_{LOW}$ ) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

$$SDADEL \leq \{t_{VD;DAT}(max) - t_r(max) - 260\text{ ns} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC+1) \times t_{I2CCLK}\}.$$

*Note:* This condition can be violated when  $NOSTRETCH = 0$ , because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.

Refer to [Table 191](#) for  $t_f$ ,  $t_r$ ,  $t_{HD;DAT}$  and  $t_{VD;DAT}$  standard values.

- After  $t_{SDADEL}$  delay, or after sending SDA output in case the slave had to stretch the clock because the data was not yet written in I2C\_TXDR register, SCL line is kept at low level during the setup time. This setup time is  $t_{SCLDEL} = (SCLDEL+1) \times t_{PRESC}$  where  $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$ .  
 $t_{SCLDEL}$  impacts the setup time  $t_{SU;DAT}$ .

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL in such a way that:

$$\{[t_r(max) + t_{SU;DAT}(min)] / [(PRESC+1) \times t_{I2CCLK}]\} - 1 \leq SCLDEL$$

Refer to [Table 191](#) for  $t_r$  and  $t_{SU;DAT}$  standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.

*Note:* At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least  $[(SDADEL + SCLDEL + 1) \times (PRESC + 1) + 1] \times t_{I2CCLK}$ , in both

transmission and reception modes. In transmission mode, if the data is not yet written in I2C\_TXDR when SDADEL counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

If NOSTRETCH = 1 in slave mode, the SCL is not stretched. Consequently the SDADEL must be programmed in such a way to guarantee also a sufficient setup time.

**Table 191. I<sup>2</sup>C-SMBus specification data setup and hold times**

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min.	Max	Min.	Max	Min.	Max	Min.	Max	
$t_{HD;DAT}$	<b>Data hold time</b>	0	-	0	-	0	-	0.3	-	μs
$t_{VD;DAT}$	<b>Data valid time</b>	-	3.45	-	0.9	-	0.45	-	-	μs
$t_{SU;DAT}$	<b>Data setup time</b>	250	-	100	-	50	-	250	-	ns
$t_r$	<b>Rise time of both SDA and SCL signals</b>	-	1000	-	300	-	120	-	1000	ns
$t_f$	<b>Fall time of both SDA and SCL signals</b>	-	300	-	300	-	120	-	300	ns

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C\_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is  $t_{SCLL} = (SCLL + 1) \times t_{PRESC}$  where  $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$ .  $t_{SCLL}$  impacts the SCL low time  $t_{LOW}$ .
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is  $t_{SCLH} = (SCLH + 1) \times t_{PRESC}$ , where  $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$ .  $t_{SCLH}$  impacts the SCL high time  $t_{HIGH}$ .

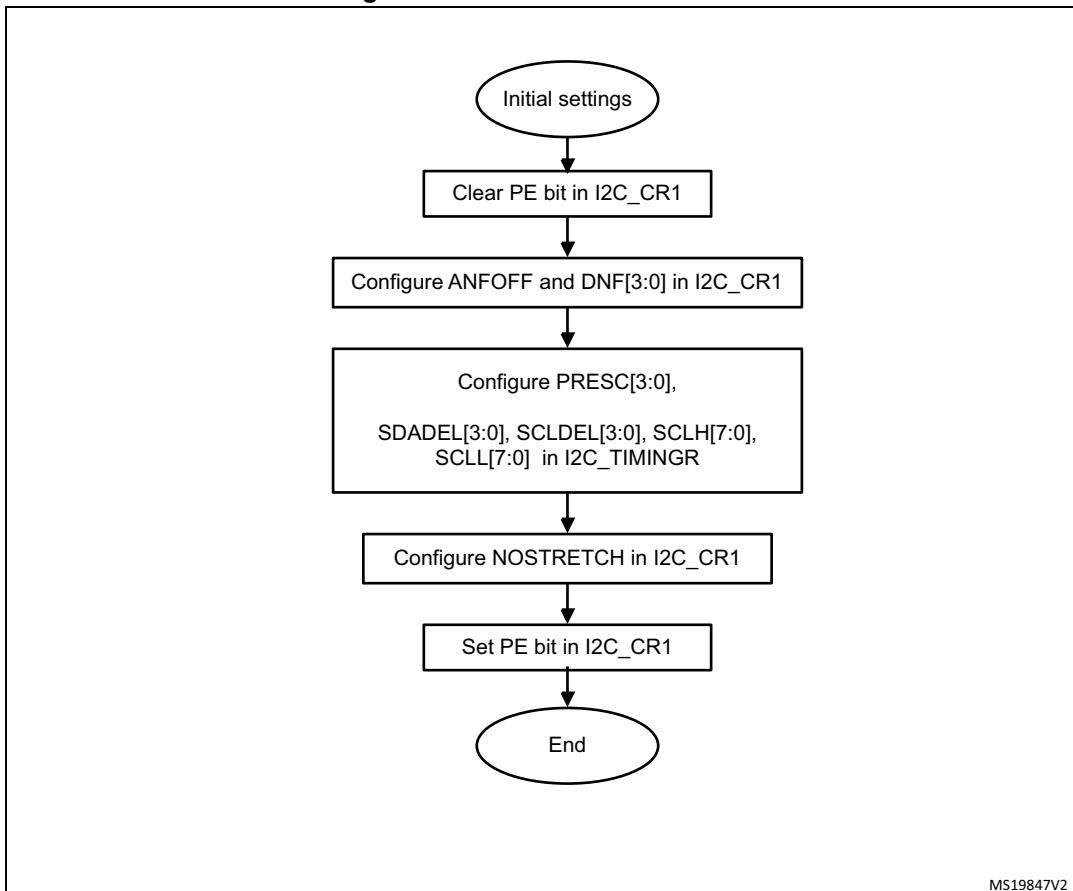
Refer to [I2C master initialization](#) for more details.

**Caution:** Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to [I2C slave initialization](#) for more details.

**Caution:** Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

Figure 293. I2C initialization flow



### 32.4.6 Software reset

A software reset can be performed by clearing the PE bit in the I2C\_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C\_CR2 register: START, STOP, NACK
2. I2C\_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C\_CR2 register: PECBYTE
2. I2C\_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least three APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

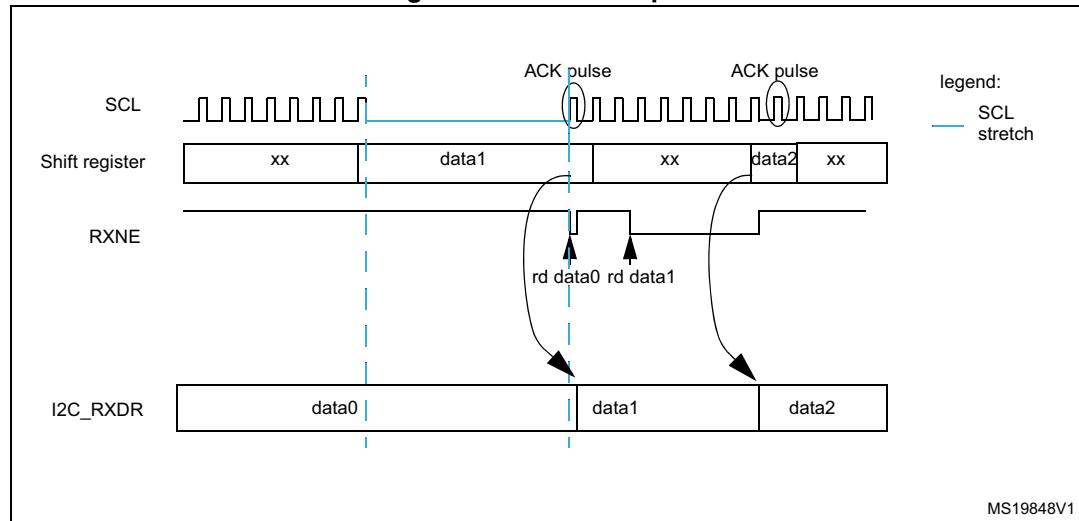
### 32.4.7 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

#### Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into I2C\_RXDR register if it is empty (RXNE = 0). If RXNE = 1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C\_RXDR is read. The stretch is inserted between the eighth and ninth SCL pulse (before the acknowledge pulse).

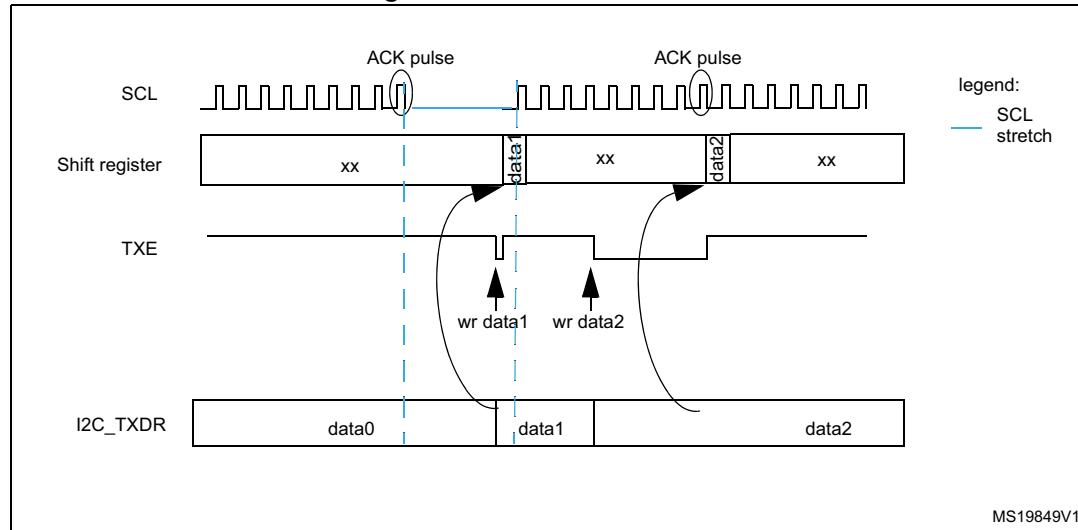
Figure 294. Data reception



## Transmission

If the I2C\_TXDR register is not empty (TXE=0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE = 1, meaning that no data is written yet in I2C\_TXDR, SCL line is stretched low until I2C\_TXDR is written. The stretch is done after the ninth SCL pulse.

Figure 295. Data transmission



## Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the SBC (slave byte control) bit in the I2C\_CR1 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C\_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C\_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES is transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.

When RELOAD=0 in master mode, the counter can be used in two modes:

- **Automatic end mode** (AUTOEND = '1' in the I2C\_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field is transferred.
- **Software end mode** (AUTOEND = '0' in the I2C\_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C\_CR2 register. This mode must be used when the master wants to send a RESTART condition.

**Caution:** The AUTOEND bit has no effect when the RELOAD bit is set.

**Table 192. I2C configuration**

Function	SBC bit	RELOAD bit	AUTOEND bit
Master Tx/Rx NBYTES + STOP	x	0	1
Master Tx/Rx + NBYTES + RESTART	x	0	0
Slave Tx/Rx all received bytes ACKed	0	x	x
Slave Rx with ACK control	1	1	x

### 32.4.8 I2C slave mode

#### I2C slave initialization

In order to work in slave mode, the user must enable at least one slave address. Two registers I2C\_OAR1 and I2C\_OAR2 are available in order to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C\_OAR1 register.  
OA1 is enabled by setting the OA1EN bit in the I2C\_OAR1 register.
- If additional slave addresses are required, the second slave address OA2 can be configured. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C\_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C\_OAR1 or I2C\_OAR2 register with OA2MSK=0.

OA2 is enabled by setting the OA2EN bit in the I2C\_OAR2 register.

- The general call address is enabled by setting the GCEN bit in the I2C\_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOSTRETCH = 1 in the I2C\_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled the user must read the ADDCODE[6:0] bits in the I2C\_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

### Slave clock stretching (NOSTRETCH = 0)

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C\_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE = 1). This stretch is released when the data is written to the I2C\_TXDR register.
- In reception when the I2C\_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C\_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during  $[(SDADEL+SCLDEL+1) \times (PRESC+1) + 1] \times t_{I2CCLK}$ .

### Slave without clock stretching (NOSTRETCH = 1)

When NOSTRETCH = 1 in the I2C\_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C\_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C\_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, he ensures that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C\_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C\_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Slave byte control mode

In order to allow byte ACK control in slave reception mode, The Slave byte control mode must be enabled by setting the SBC bit in the I2C\_CR1 register. This is required to be compliant with SMBus standards.

The Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD = 1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and ninth SCL pulses. The user can read the data from the I2C\_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C\_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

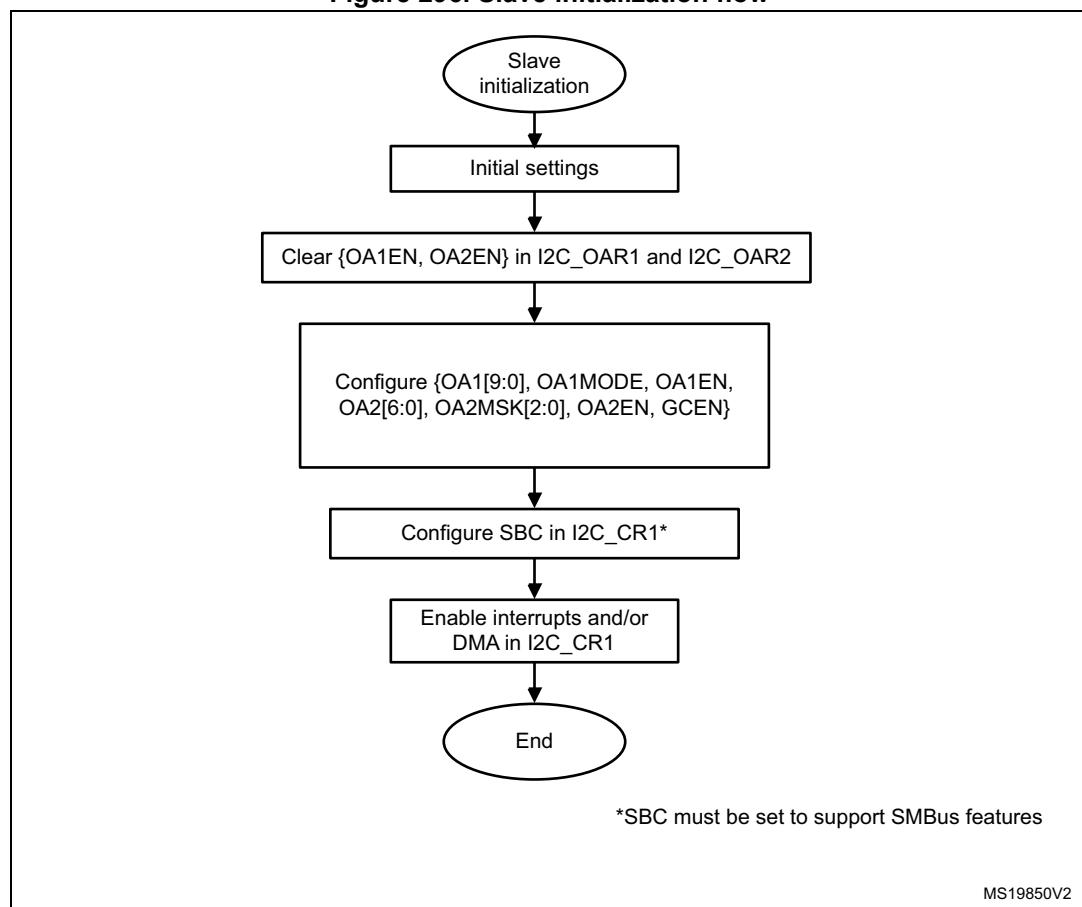
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

**Note:** *The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR = 1.*

*The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.*

**Caution:** The Slave byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

**Figure 296. Slave initialization flow**



### Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C\_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C\_CR1 register.

The TXIS bit is cleared when the I2C\_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C\_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C\_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C\_CR1 register, the STOPF flag is set in the I2C\_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to '0'. In this case, If TXE = 0 when the slave address is received (ADDR = 1), the user can choose either to send the content of the I2C\_TXDR register as the first data byte, or to flush the I2C\_TXDR register by setting the TXE bit in order to program a new data byte.

In Slave byte control mode (SBC = 1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

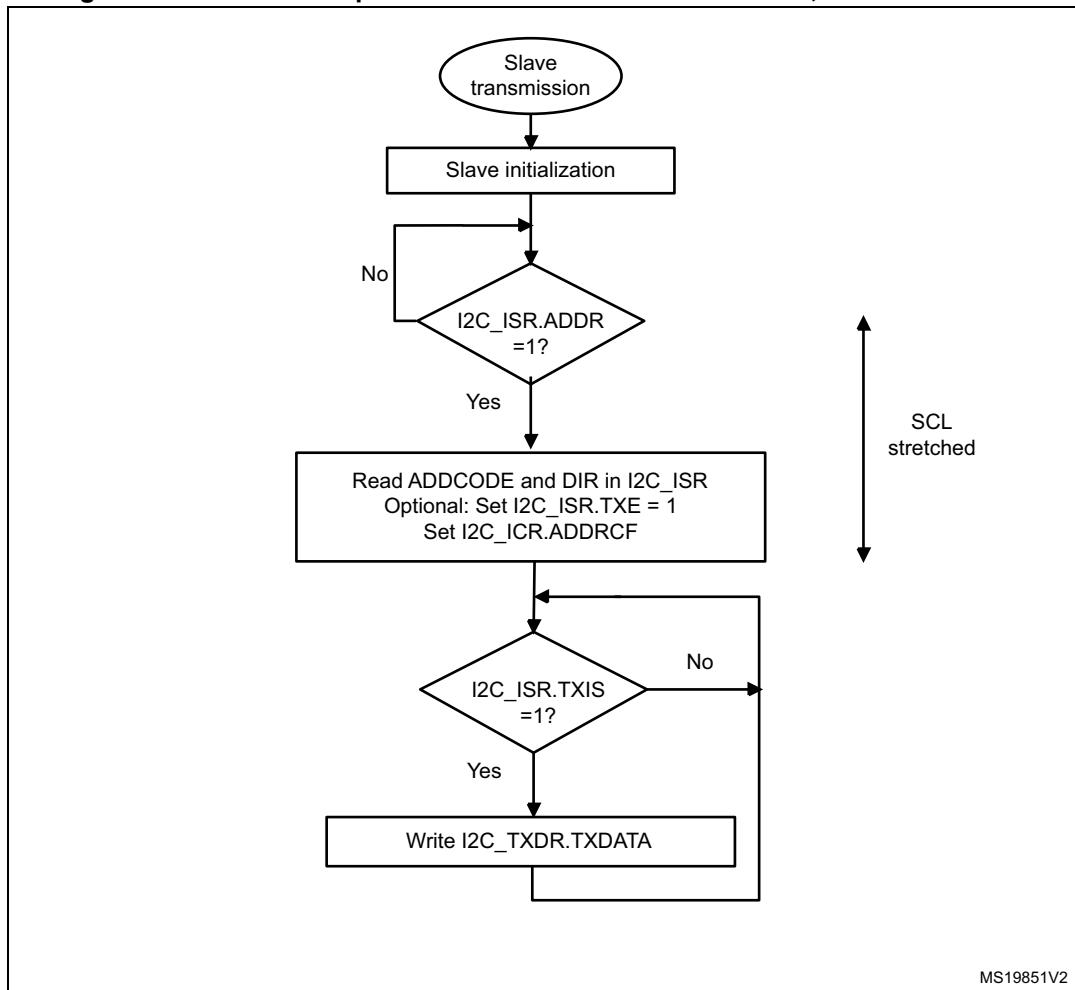
**Caution:** When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C\_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C\_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C\_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

If a TXIS event is needed, (transmit interrupt or transmit DMA request), the user must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.

Figure 297. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0



MS19851V2

Figure 298. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1

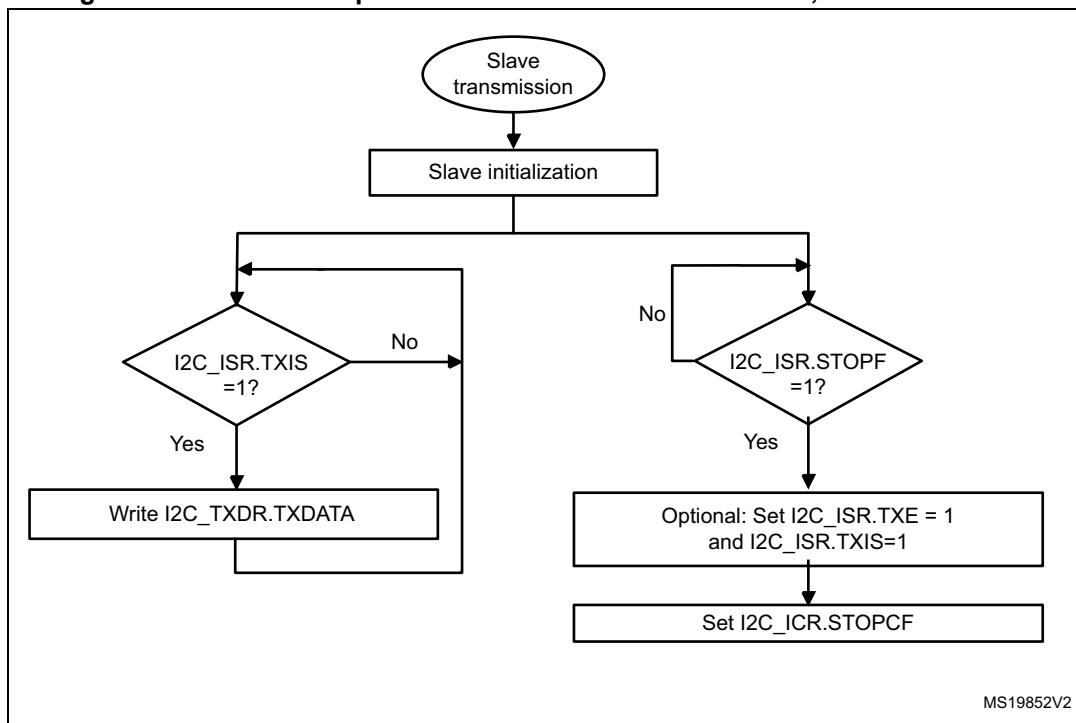
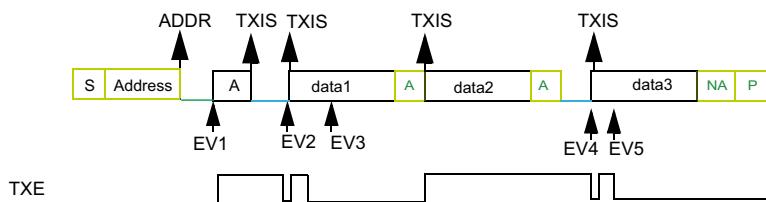


Figure 299. Transfer bus diagrams for I2C slave transmitter (mandatory events only)

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



legend:

- transmission
- reception
- SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF

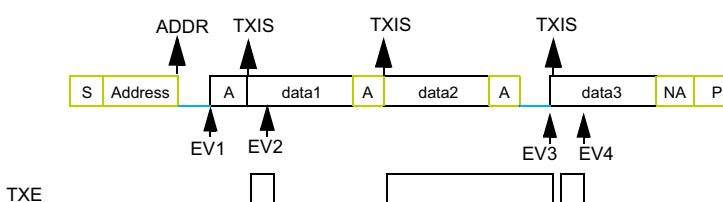
EV2: TXIS ISR: wr data1

EV3: TXIS ISR: wr data2

EV4: TXIS ISR: wr data3

EV5: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



legend :

- transmission
- reception
- SCL stretch

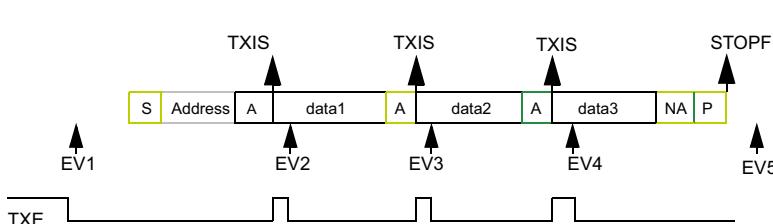
EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF

EV2: TXIS ISR: wr data2

EV3: TXIS ISR: wr data3

EV4: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:



legend:

- transmission
- reception
- SCL stretch

EV1: wr data1

EV2: TXIS ISR: wr data2

EV3: TXIS ISR: wr data3

EV4: TXIS ISR: wr data4 (not sent)

EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MS19853V2

### Slave receiver

RXNE is set in I2C\_ISR when the I2C\_RXDR is full, and generates an interrupt if RXIE is set in I2C\_CR1. RXNE is cleared when I2C\_RXDR is read.

When a STOP is received and STOPIE is set in I2C\_CR1, STOPF is set in I2C\_ISR and an interrupt is generated.

Figure 300. Transfer sequence flow for slave receiver with NOSTRETCH = 0

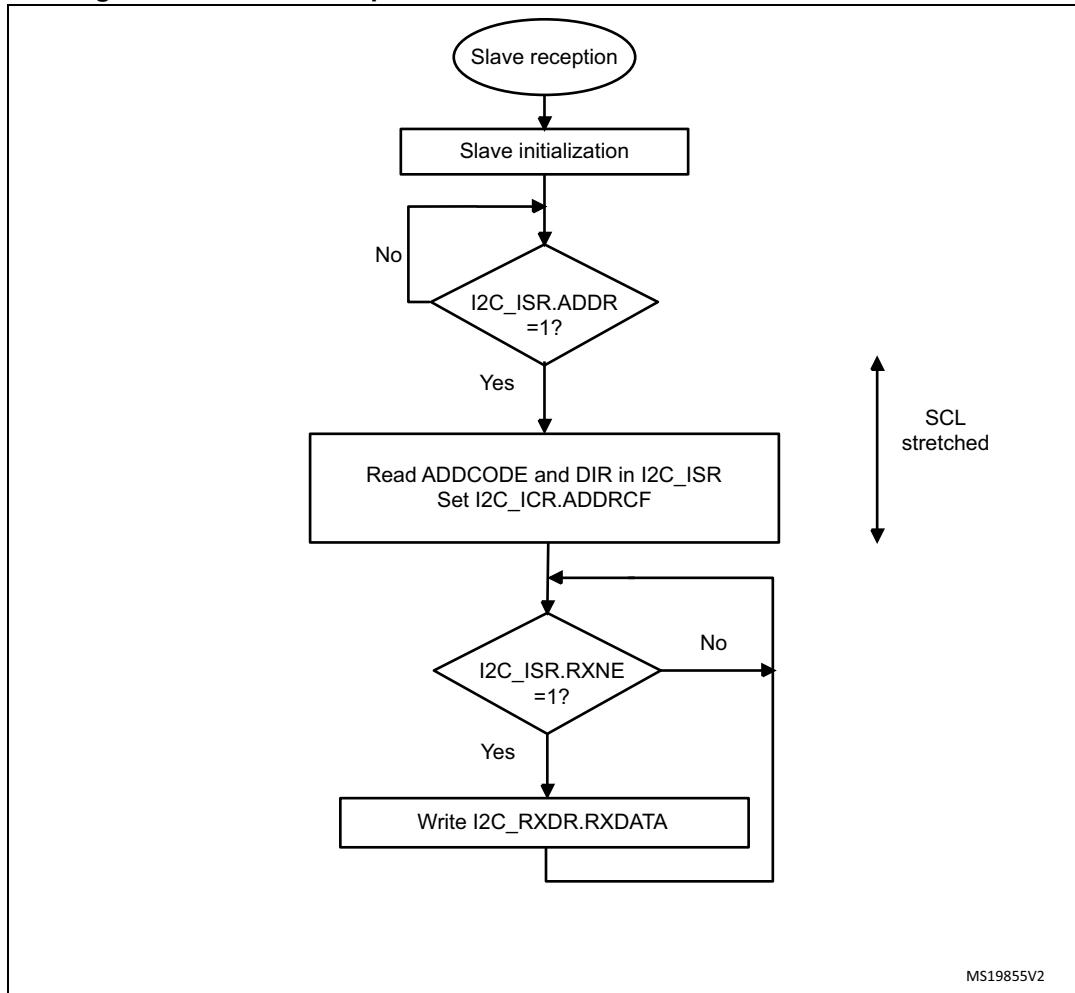


Figure 301. Transfer sequence flow for slave receiver with NOSTRETCH = 1

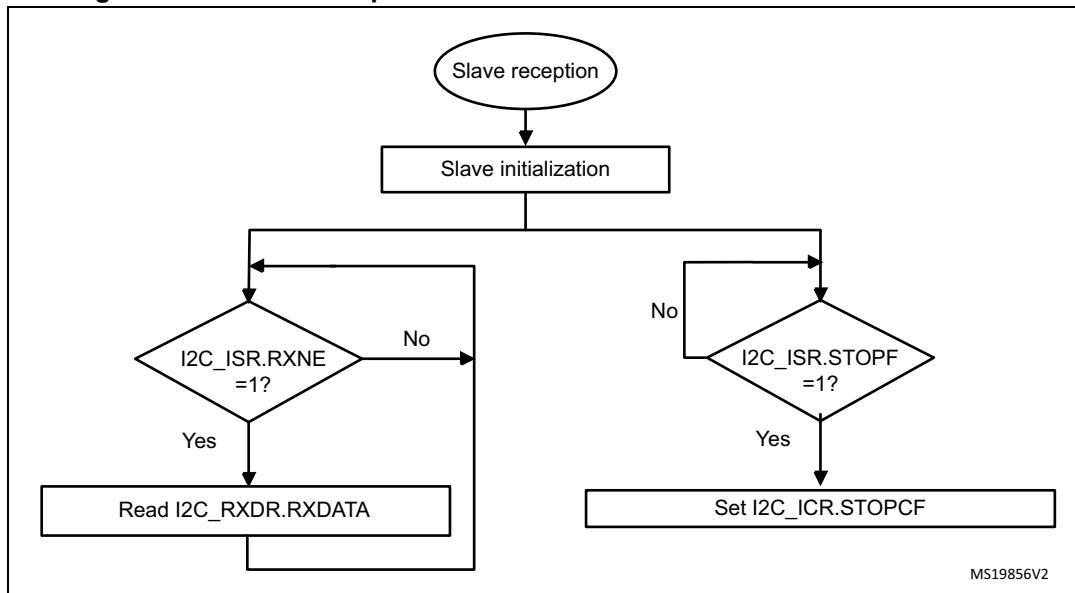
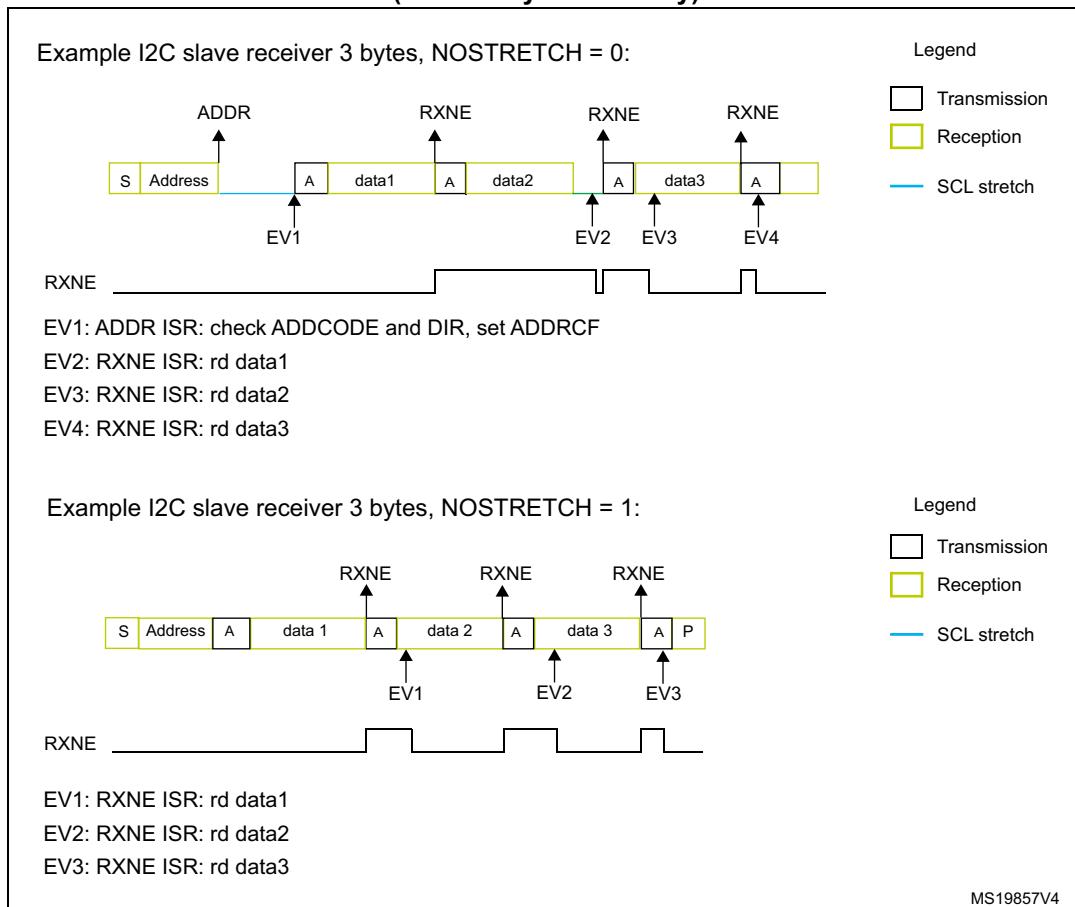


Figure 302. Transfer bus diagrams for I2C slave receiver (mandatory events only)



### 32.4.9 I2C master mode

#### I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C\_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C\_TIMINGR content in the I2C Configuration window.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a  $t_{SYNC1}$  delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2C\_TIMINGR register.

The I2C detects its own SCL high level after a  $t_{SYNC2}$  delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter is reached reaches the value programmed in the SCLH[7:0] bits in the I2C\_TIMINGR register.

Consequently the master clock period is:

$$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$$

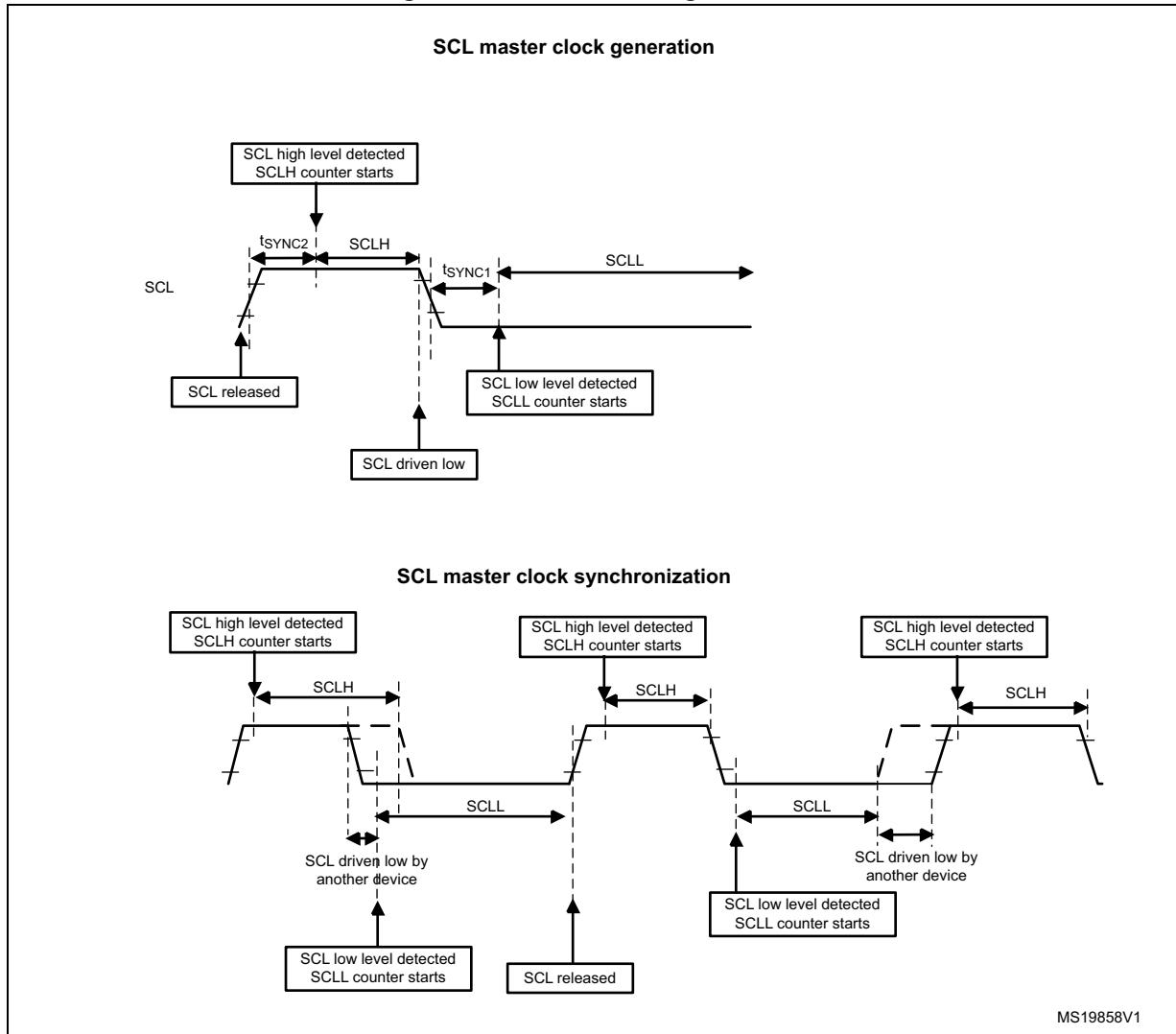
The duration of  $t_{SYNC1}$  depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF  $\times t_{I2CCLK}$
- Delay due to SCL synchronization with I2CCLK clock (two to three I2CCLK periods)

The duration of  $t_{SYNC2}$  depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF  $\times t_{I2CCLK}$
- Delay due to SCL synchronization with I2CCLK clock (two to three I2CCLK periods)

Figure 303. Master clock generation



**Caution:** To be I<sup>2</sup>C or SMBus compliant, the master clock must respect the timings given in the table below.

**Table 193. I<sup>2</sup>C-SMBus specification clock timings**

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$f_{SCL}$	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	$\mu$ s
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
$t_{BUF}$	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
$t_{LOW}$	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
$t_{HIGH}$	Period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	ns
$t_r$	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
$t_f$	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

**Note:** *SCLL is also used to generate the  $t_{BUF}$  and  $t_{SU:STA}$  timings.*

*SCLH is also used to generate the  $t_{HD:STA}$  and  $t_{SU:STO}$  timings.*

Refer to [Section 32.4.10: I2C\\_TIMINGR register configuration examples](#) for examples of I2C\_TIMINGR settings vs. I2CCLK frequency.

### Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C\_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD\_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C\_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of  $t_{BUF}$ .

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

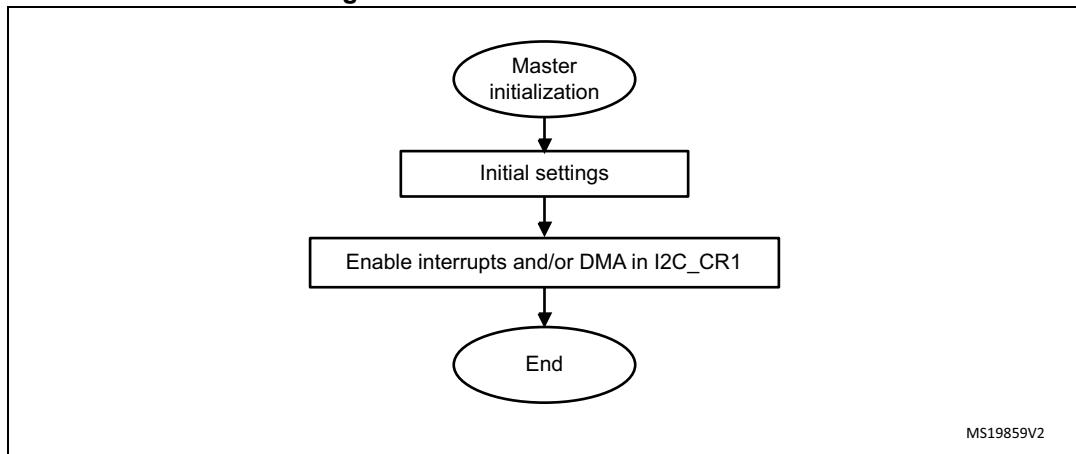
**Note:** The *START* bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value. The *START* bit is also reset by hardware if an arbitration loss occurs.

In 10-bit addressing mode, when the slave address first 7 bits are NACKed by the slave, the master re-launches automatically the slave address transmission until ACK is received. In this case ADDRCF must be set if a NACK is received from the slave, in order to stop sending the slave address.

If the I2C is addressed as a slave (ADDR = 1) while the *START* bit is set, the I2C switches to slave mode and the *START* bit is cleared, when the ADDRCF bit is set.

**Note:** The same procedure is applied for a repeated start condition. In this case BUSY = 1.

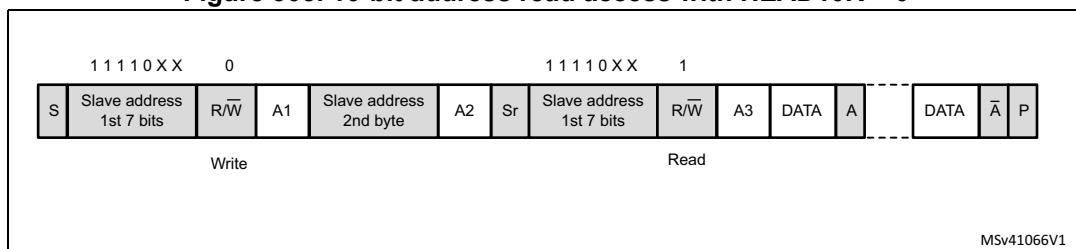
**Figure 304. Master initialization flow**



#### Initialization of a master receiver addressing a 10-bit address slave

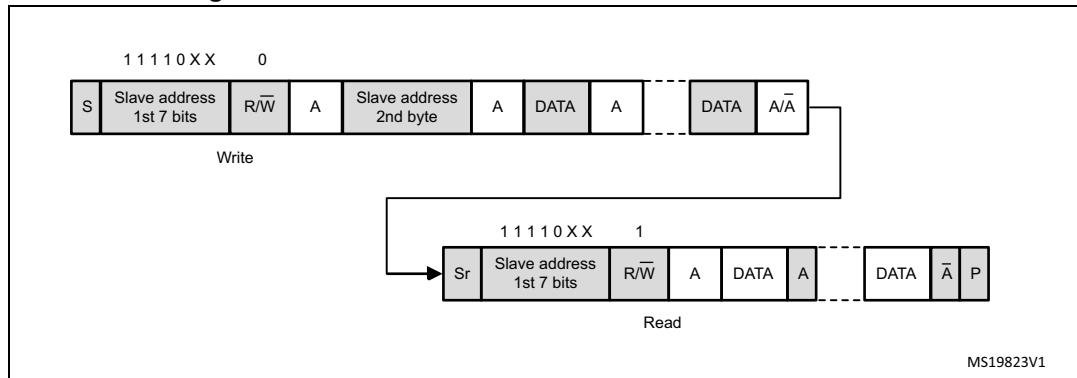
- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C\_CR2 register. In this case the master automatically sends the following complete sequence after the *START* bit is set: (Re)Start + Slave address 10-bit header Write + Slave address second byte + REStart + Slave address 10-bit header Read

**Figure 305. 10-bit address read access with HEAD10R = 0**



- If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated start is set with the 10 bit slave address configured with HEAD10R = 1. In this case the master sends this sequence: ReStart + Slave address 10-bit header Read.

Figure 306. 10-bit address read access with HEAD10R = 1



### Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C\_CR1 register. The flag is cleared when the I2C\_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C\_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND = 1), a STOP is automatically sent.
  - In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low in order to perform software actions:
 

A RESTART condition can be requested by setting the START bit in the I2C\_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.

A STOP condition can be requested by setting the STOP bit in the I2C\_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C\_ISR register, and an interrupt is generated if the NACKIE bit is set.

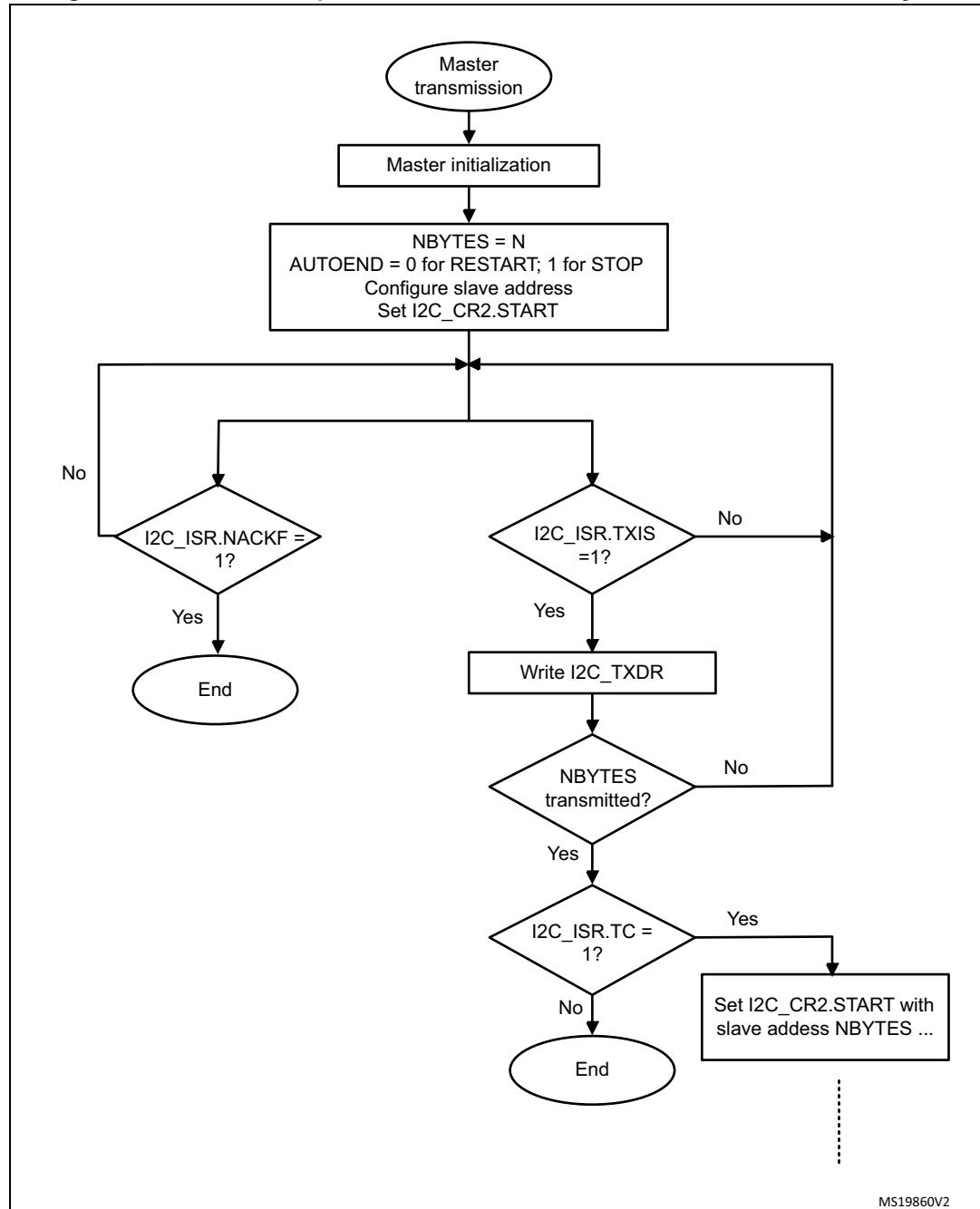
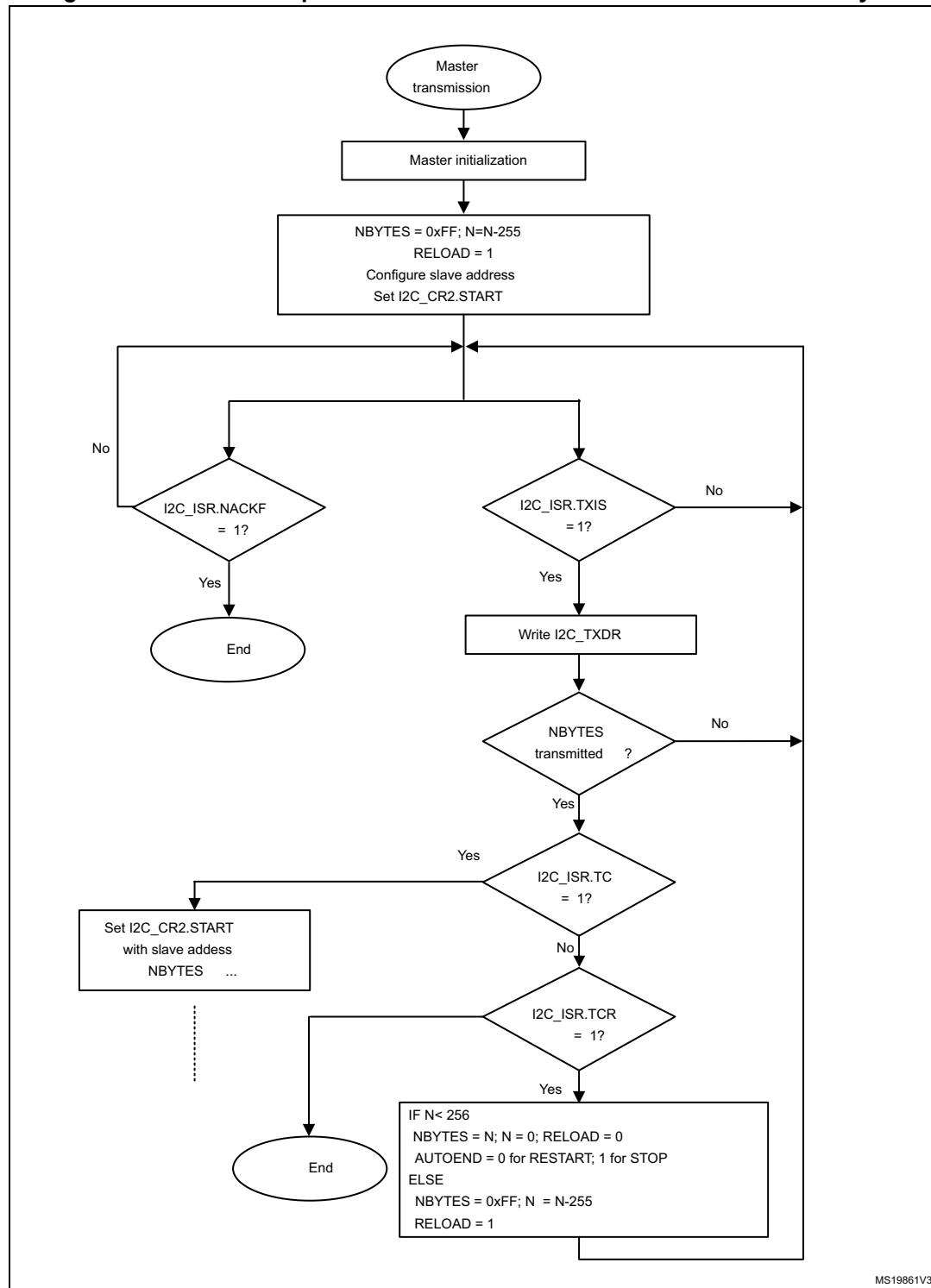
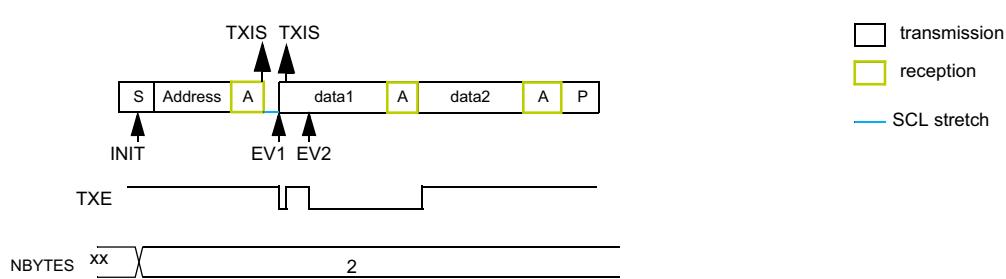
Figure 307. Transfer sequence flow for I2C master transmitter for  $N \leq 255$  bytes

Figure 308. Transfer sequence flow for I2C master transmitter for  $N > 255$  bytes

MS19861V3

**Figure 309. Transfer bus diagrams for I2C master transmitter (mandatory events only)**

Example I2C master transmitter 2 bytes, automatic end mode (STOP)

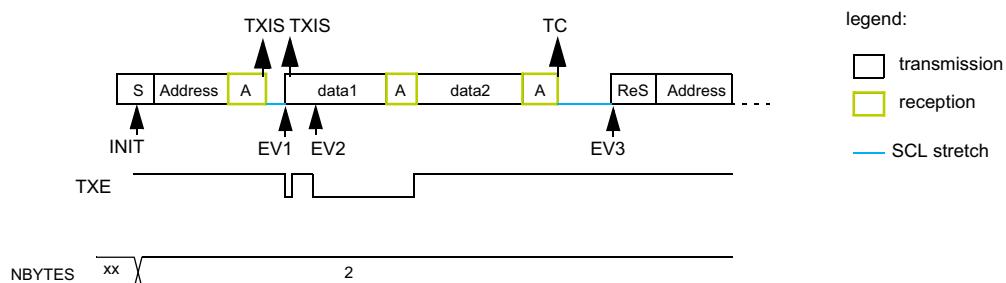


INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

Example I2C master transmitter 2 bytes, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

EV3: TC ISR: program Slave address, program NBYTES = N, set START

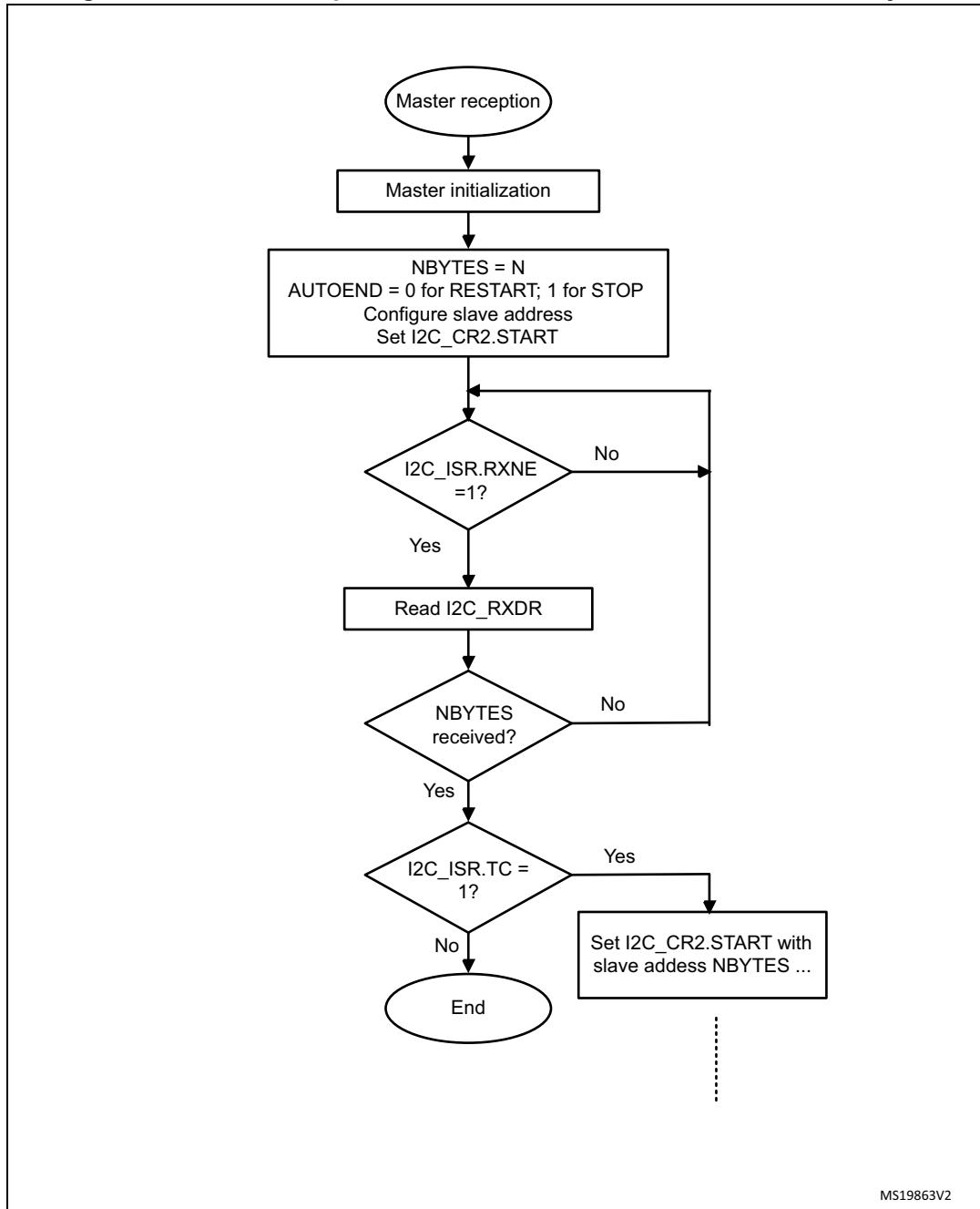
MS19862V2

## Master receiver

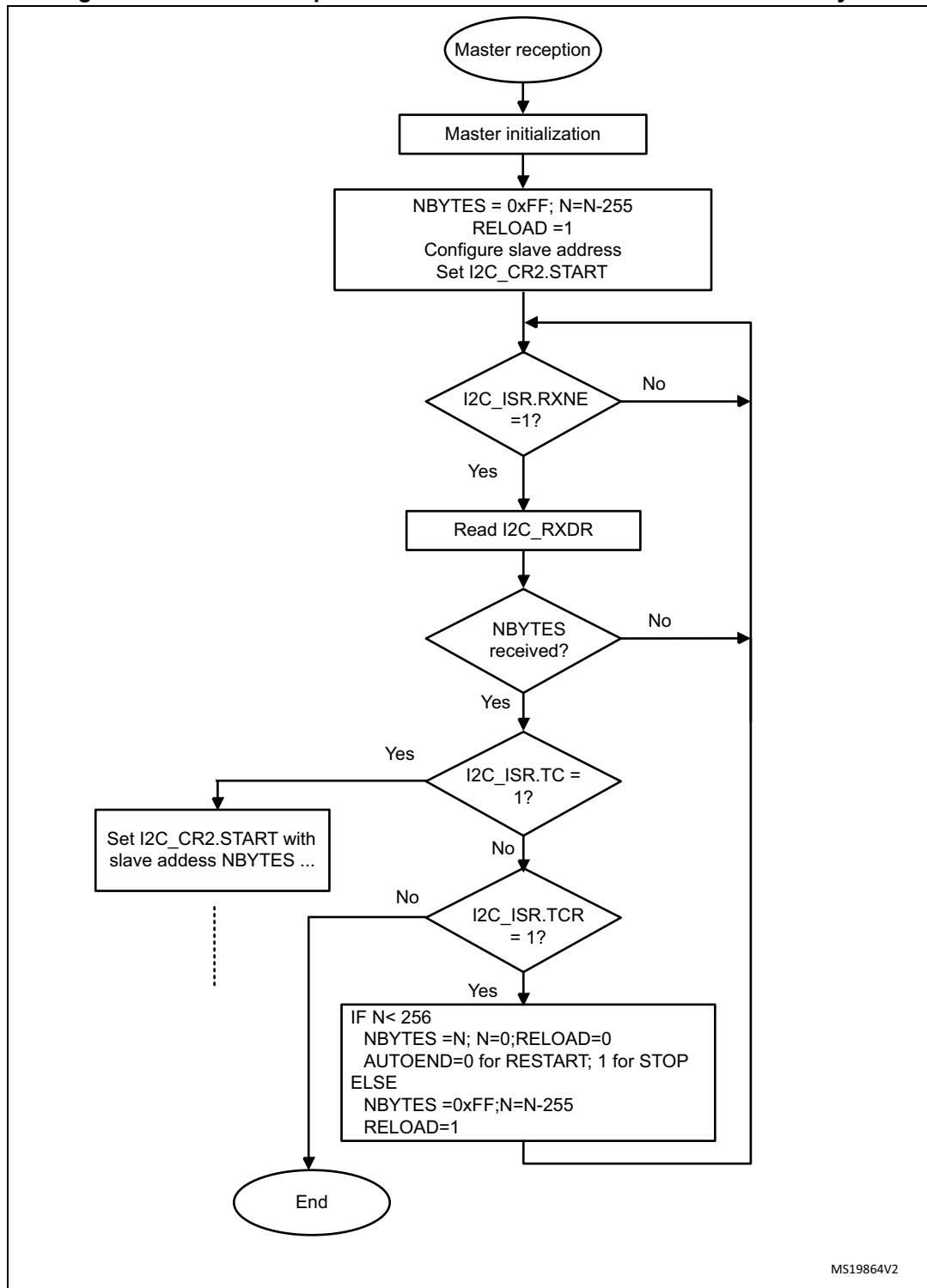
In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C\_CR1 register. The flag is cleared when I2C\_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C\_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:  
A RESTART condition can be requested by setting the START bit in the I2C\_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.  
A STOP condition can be requested by setting the STOP bit in the I2C\_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

Figure 310. Transfer sequence flow for I2C master receiver for  $N \leq 255$  bytes

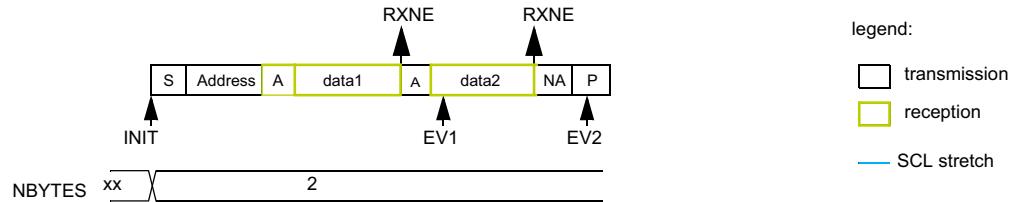
MS19863V2

Figure 311. Transfer sequence flow for I2C master receiver for  $N > 255$  bytes

MS19864V2

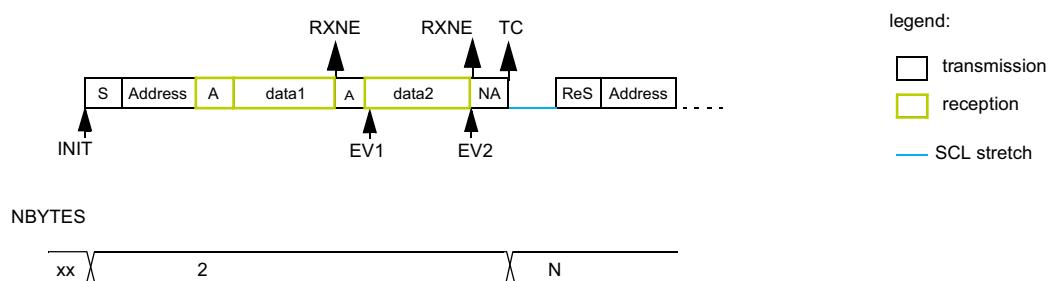
**Figure 312. Transfer bus diagrams for I2C master receiver (mandatory events only)**

Example I2C master receiver 2 bytes, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START  
 EV1: RXNE ISR: rd data1  
 EV2: RXNE ISR: rd data2

Example I2C master receiver 2 bytes, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START  
 EV1: RXNE ISR: rd data1  
 EV2: RXNE ISR: read data2  
 EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19865V1

### 32.4.10 I2C\_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C\_TIMINGR to obtain timings compliant with the I<sup>2</sup>C specification. In order to get more accurate configuration values, the STM32CubeMX tool (I2C Configuration window) must be used.

**Table 194. Examples of timing settings for  $f_{I2CCLK} = 8$  MHz**

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC	1	1	0	0
SCLL	0xC7	0x13	0x9	0x6
$t_{SCLL}$	$200 \times 250 \text{ ns} = 50 \mu\text{s}$	$20 \times 250 \text{ ns} = 5.0 \mu\text{s}$	$10 \times 125 \text{ ns} = 1250 \text{ ns}$	$7 \times 125 \text{ ns} = 875 \text{ ns}$
SCLH	0xC3	0xF	0x3	0x3
$t_{SCLH}$	$196 \times 250 \text{ ns} = 49 \mu\text{s}$	$16 \times 250 \text{ ns} = 4.0 \mu\text{s}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$
$t_{SCL}^{(1)}$	$\sim 100 \mu\text{s}^{(2)}$	$\sim 10 \mu\text{s}^{(2)}$	$\sim 2500 \text{ ns}^{(3)}$	$\sim 2000 \text{ ns}^{(4)}$
SDADEL	0x2	0x2	0x1	0x0
$t_{SDADEL}$	$2 \times 250 \text{ ns} = 500 \text{ ns}$	$2 \times 250 \text{ ns} = 500 \text{ ns}$	$1 \times 125 \text{ ns} = 125 \text{ ns}$	0 ns
SCLDEL	0x4	0x4	0x3	0x1
$t_{SCLDEL}$	$5 \times 250 \text{ ns} = 1250 \text{ ns}$	$5 \times 250 \text{ ns} = 1250 \text{ ns}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$	$2 \times 125 \text{ ns} = 250 \text{ ns}$

1. SCL period  $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to SCL internal detection delay. Values provided for  $t_{SCL}$  are examples only.

2.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 1000 \text{ ns}$ .

3.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 750 \text{ ns}$ .

4.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 655 \text{ ns}$ .

**Table 195. Examples of timings settings for  $f_{I2CCLK} = 16$  MHz**

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
$t_{SCLL}$	$200 \times 250 \text{ ns} = 50 \mu\text{s}$	$20 \times 250 \text{ ns} = 5.0 \mu\text{s}$	$10 \times 125 \text{ ns} = 1250 \text{ ns}$	$5 \times 62.5 \text{ ns} = 312.5 \text{ ns}$
SCLH	0xC3	0xF	0x3	0x2
$t_{SCLH}$	$196 \times 250 \text{ ns} = 49 \mu\text{s}$	$16 \times 250 \text{ ns} = 4.0 \mu\text{s}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$	$3 \times 62.5 \text{ ns} = 187.5 \text{ ns}$
$t_{SCL}^{(1)}$	$\sim 100 \mu\text{s}^{(2)}$	$\sim 10 \mu\text{s}^{(2)}$	$\sim 2500 \text{ ns}^{(3)}$	$\sim 1000 \text{ ns}^{(4)}$
SDADEL	0x2	0x2	0x2	0x0
$t_{SDADEL}$	$2 \times 250 \text{ ns} = 500 \text{ ns}$	$2 \times 250 \text{ ns} = 500 \text{ ns}$	$2 \times 125 \text{ ns} = 250 \text{ ns}$	0 ns
SCLDEL	0x4	0x4	0x3	0x2
$t_{SCLDEL}$	$5 \times 250 \text{ ns} = 1250 \text{ ns}$	$5 \times 250 \text{ ns} = 1250 \text{ ns}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$	$3 \times 62.5 \text{ ns} = 187.5 \text{ ns}$

1. SCL period  $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to SCL internal detection delay. Values provided for  $t_{SCL}$  are examples only.

2.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 1000$  ns.
3.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 750$  ns.
4.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 500$  ns.

### 32.4.11 SMBus specific features

This section is relevant only when the SMBus feature is supported. Refer to [Section 32.3: I2C implementation](#).

#### Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I<sup>2</sup>C principles of operation. The SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBus specification (<http://smbus.org>).

The system management bus specification refers to three types of devices

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.

#### Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols must be implemented by the user software.

For more details on these protocols, refer to SMBus specification (<http://smbus.org>).

#### Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C\_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus address resolution protocol, refer to SMBus specification (<http://smbus.org>).

### Received command and data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C\_CR1 register. Refer to [Slave byte control mode](#) for more details.

### Host notify protocol

This peripheral supports the host notify protocol by setting the SMBHEN bit in the I2C\_CR1 register. In this case the host acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

### SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the alert response address.

When configured as a slave device(SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C\_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C\_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

*If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0.*

### Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the  $C(x) = x_8 + x^2 + x + 1$  CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match with the hardware calculated PEC.

## Timeouts

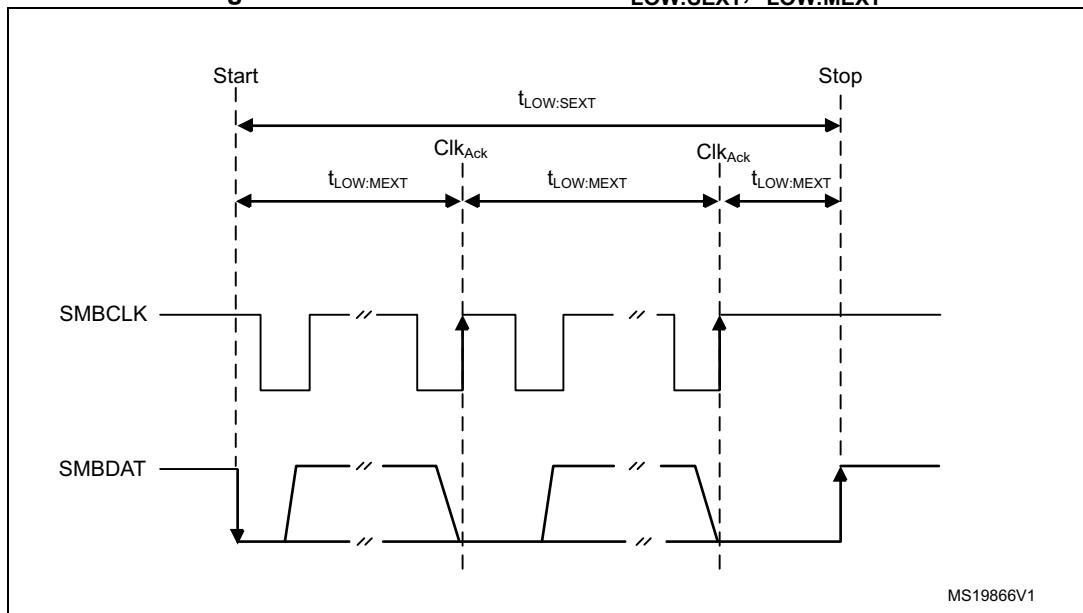
This peripheral embeds hardware timers in order to be compliant with the three timeouts defined in SMBus specification.

Table 196. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}^{(1)}$	Cumulative clock low extend time (slave device)	-	25	
$t_{LOW:MEXT}^{(2)}$	Cumulative clock low extend time (master device)	-	10	

1.  $t_{LOW:SEXT}$  is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another slave device or the master also extends the clock causing the combined clock low extend time to be greater than  $t_{LOW:SEXT}$ . Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.
2.  $t_{LOW:MEXT}$  is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master also extends the clock causing the combined clock low time to be greater than  $t_{LOW:MEXT}$  on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

Figure 313. Timeout intervals for  $t_{LOW:SEXT}$ ,  $t_{LOW:MEXT}$



### Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for  $t_{IDLE}$  greater than  $t_{HIGH,MAX}$ . (refer to [Table 191](#))

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

### 32.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to [Section 32.3: I2C implementation](#).

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

#### Received command and data acknowledge control (Slave mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave byte control mode must be enabled by setting the SBC bit in the I2C\_CR1 register. Refer to [Slave byte control mode](#) for more details.

#### Specific address (Slave mode)

The specific SMBus addresses must be enabled if needed. Refer to [Bus idle detection](#) for more details.

- The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C\_CR1 register.
- The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C\_CR1 register.
- The alert response address (0b0001100) is enabled by setting the ALERTEN bit in the I2C\_CR1 register.

#### Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C\_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2C\_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTES - 1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

**Caution:** Changing the PECEN configuration is not allowed when the I2C is enabled.

**Table 197. SMBus with PEC configuration**

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Master Tx/Rx NBYTES + PEC+ STOP	x	0	1	1
Master Tx/Rx NBYTES + PEC + ReSTART	x	0	0	1
Slave Tx/Rx with PEC	1	0	x	1

### Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the I2C\_TIMEOUTTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

- $t_{TIMEOUT}$  check

In order to enable the  $t_{TIMEOUT}$  check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the  $t_{TIMEOUT}$  parameter. The TIDLE bit must be configured to '0' in order to detect the SCL low level timeout.

Then the timer is enabled by setting the TIMOUTEN in the I2C\_TIMEOUTTR register.

If SCL is tied low for a time greater than  $(TIMEOUTA+1) \times 2048 \times t_{I2CCLK}$ , the TIMEOUT flag is set in the I2C\_ISR register.

Refer to [Table 198](#).

**Caution:** Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMOUTEN bit is set.

- $t_{LOW:SEXT}$  and  $t_{LOW:MEXT}$  check

Depending on if the peripheral is configured as a master or as a slave, The 12-bit TIMEOUTB timer must be configured in order to check  $t_{LOW:SEXT}$  for a slave and  $t_{LOW:MEXT}$  for a master. As the standard specifies only a maximum, the user can choose the same value for the both.

Then the timer is enabled by setting the TEXTEN bit in the I2C\_TIMEOUTTR register.

If the SMBus peripheral performs a cumulative SCL stretch for a time greater than  $(TIMEOUTB+1) \times 2048 \times t_{I2CCLK}$ , and in the timeout interval described in [Bus idle detection](#) section, the TIMEOUT flag is set in the I2C\_ISR register.

Refer to [Table 199](#)

**Caution:** Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

### Bus idle detection

In order to enable the  $t_{IDLE}$  check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the  $t_{IDLE}$  parameter. The TIDLE bit must be configured to '1' in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMOUTEN bit in the I2C\_TIMEOUTTR register.

If both the SCL and SDA lines remain high for a time greater than  $(TIMEOUTA+1) \times 4 \times t_{I2CCLK}$ , the TIMEOUT flag is set in the I2C\_ISR register.

Refer to [Table 200](#).

**Caution:** Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMOUTEN is set.

### 32.4.13 SMBus: I2C\_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to [Section 32.3: I2C implementation](#).

- Configuring the maximum duration of  $t_{TIMEOUT}$  to 25 ms:

**Table 198. Examples of TIMEOUTA settings for various I2CCLK frequencies  
(max  $t_{TIMEOUT} = 25$  ms)**

$f_{I2CCLK}$	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIMEOUT}$
8 MHz	0x61	0	1	$98 \times 2048 \times 125$ ns = 25 ms
16 MHz	0xC3	0	1	$196 \times 2048 \times 62.5$ ns = 25 ms

- Configuring the maximum duration of  $t_{LOW:SEXT}$  and  $t_{LOW:MEXT}$  to 8 ms:

**Table 199. Examples of TIMEOUTB settings for various I2CCLK frequencies**

$f_{I2CCLK}$	TIMEOUTB[11:0] bits	TEXTEN bit	$t_{LOW:EXT}$
8 MHz	0x1F	1	$32 \times 2048 \times 125$ ns = 8 ms
16 MHz	0x3F	1	$64 \times 2048 \times 62.5$ ns = 8 ms

- Configuring the maximum duration of  $t_{IDLE}$  to 50  $\mu$ s

**Table 200. Examples of TIMEOUTA settings for various I2CCLK frequencies  
(max  $t_{IDLE} = 50$   $\mu$ s)**

$f_{I2CCLK}$	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIDLE}$
8 MHz	0x63	1	1	$100 \times 4 \times 125$ ns = 50 $\mu$ s
16 MHz	0xC7	1	1	$200 \times 4 \times 62.5$ ns = 50 $\mu$ s

### 32.4.14 SMBus slave mode

This section is relevant only when the SMBus feature is supported. Refer to [Section 32.3: I2C implementation](#).

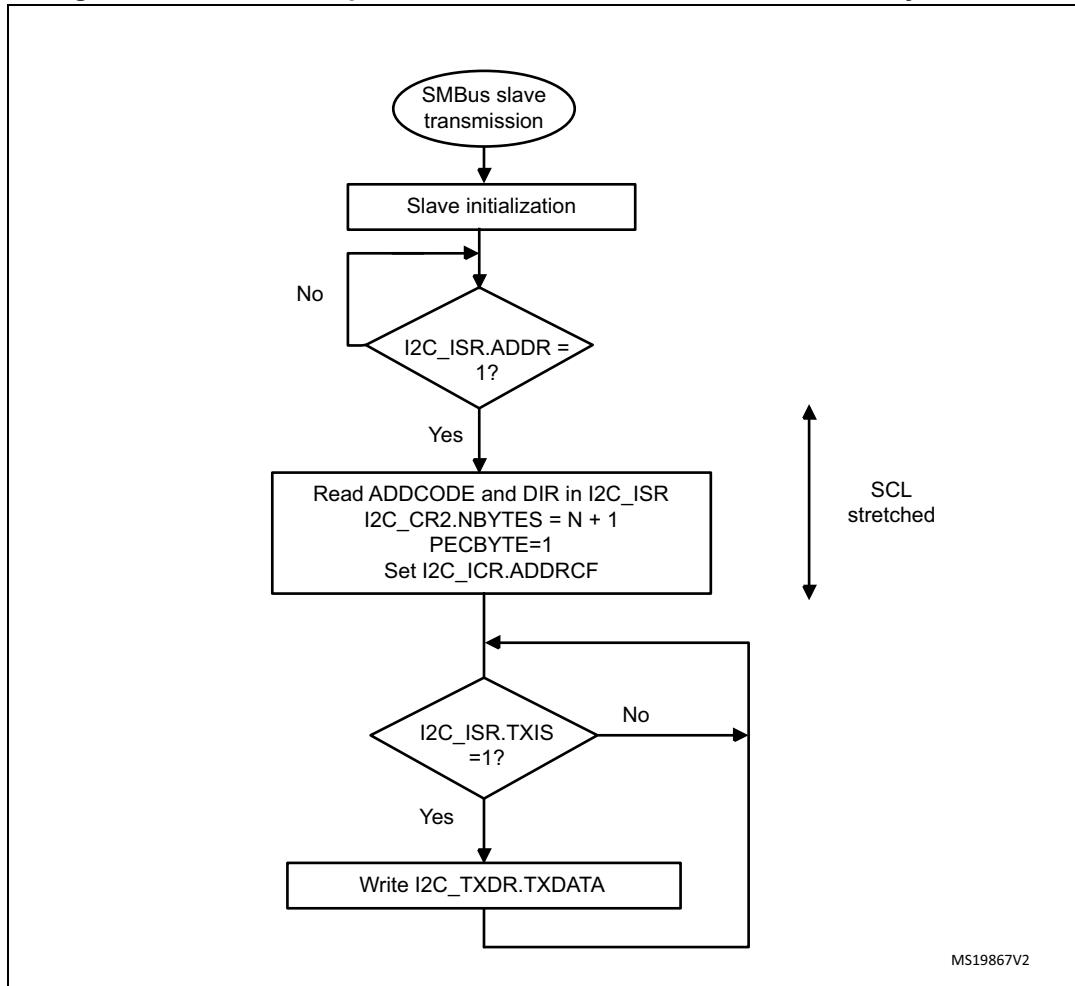
In addition to I2C slave transfer management (refer to [Section 32.4.8: I2C slave mode](#)) some additional software flows are provided to support the SMBus.

#### SMBus slave transmitter

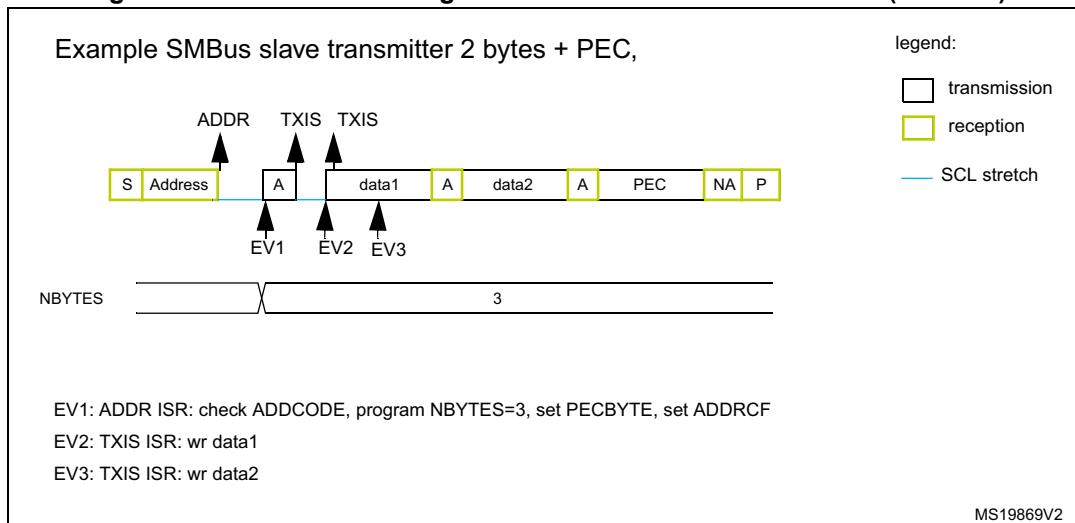
When the IP is used in SMBus, SBC must be programmed to '1' in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts is NBYTES - 1 and the content of the I2C\_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES - 1 data transfer.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 314. Transfer sequence flow for SMBus slave transmitter N bytes + PEC



**Figure 315. Transfer bus diagrams for SMBus slave transmitter (SBC = 1)**



### SMBus Slave receiver

When the I2C is used in SMBus mode, SBC must be programmed to '1' in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to [Slave byte control mode](#) for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES - 1 data have been received, the next received byte is compared with the internal I2C\_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C\_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

If no ACK software control is needed, the user can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES - 1 are received, the next received byte is checked as being the PEC.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 316. Transfer sequence flow for SMBus slave receiver N bytes + PEC

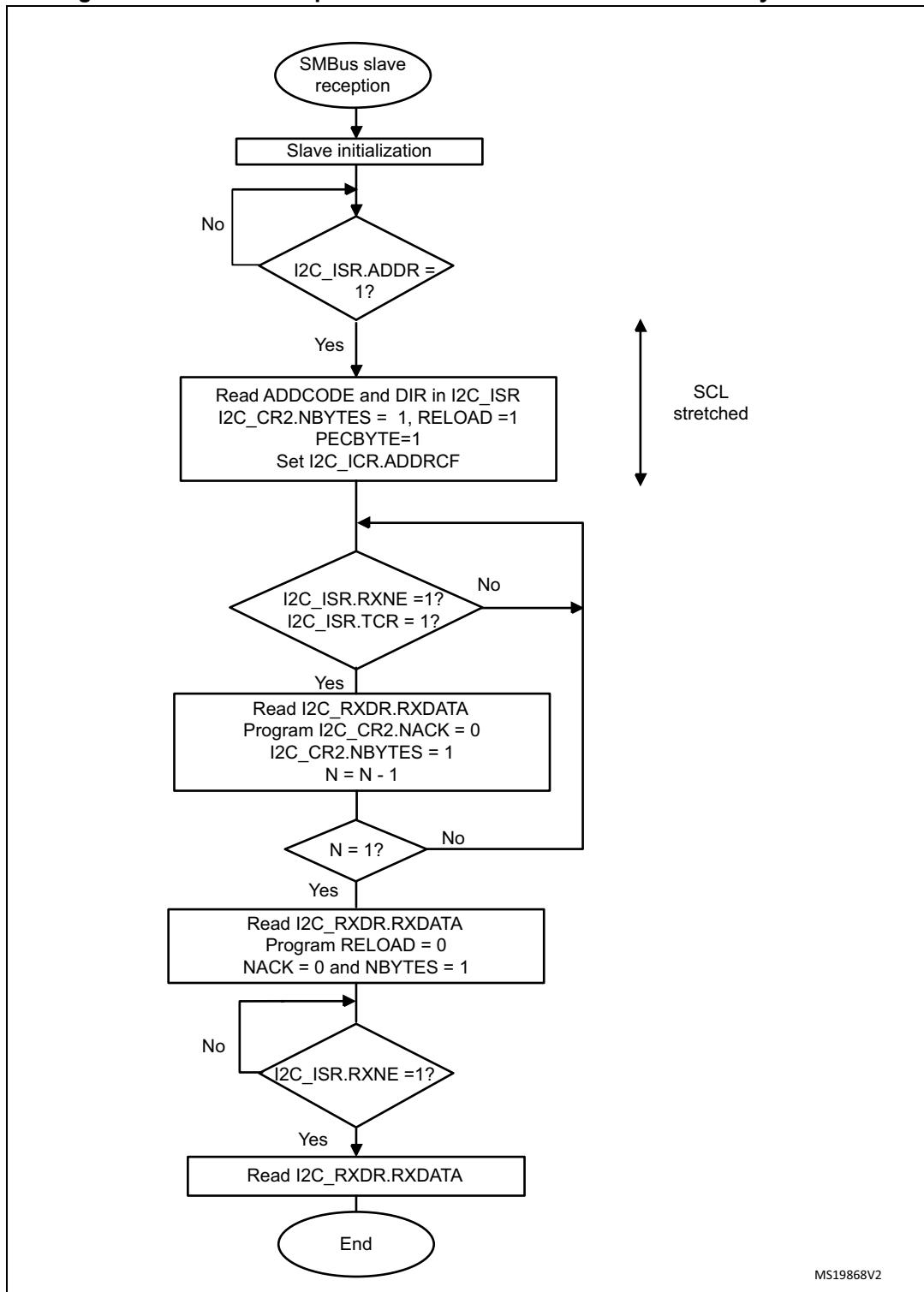
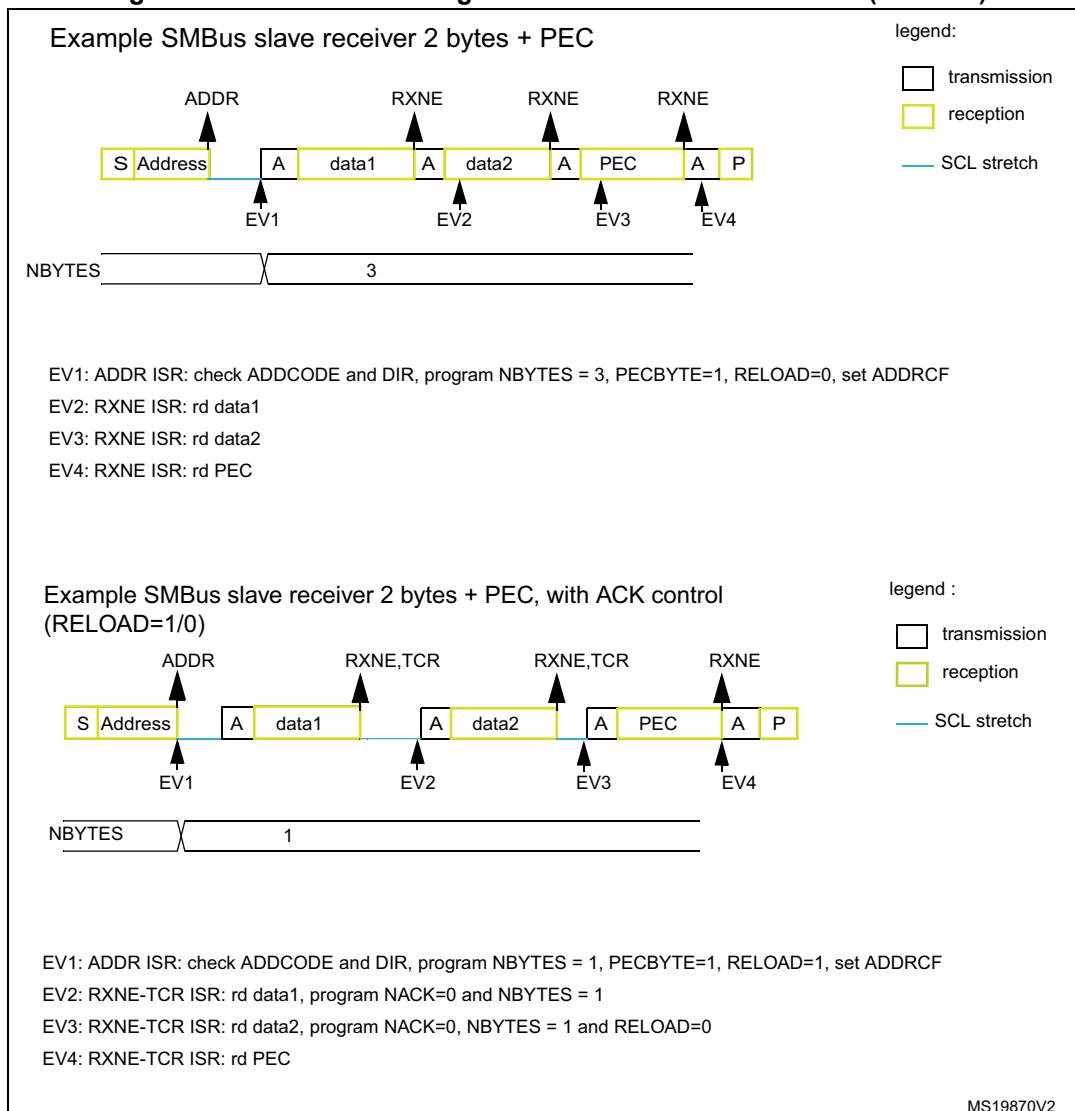


Figure 317. Bus transfer diagrams for SMBus slave receiver (SBC = 1)



MS19870V2

This section is relevant only when the SMBus feature is supported. Refer to [Section 32.3: I2C implementation](#).

In addition to I2C master transfer management (refer to [Section 32.4.9: I2C master mode](#)), some additional software flows are provided to support the SMBus.

### SMBus master transmitter

When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES - 1. So if the PECBYTE bit is set when NBYTES = 0x1, the content of the I2C\_PECR register is automatically transmitted.

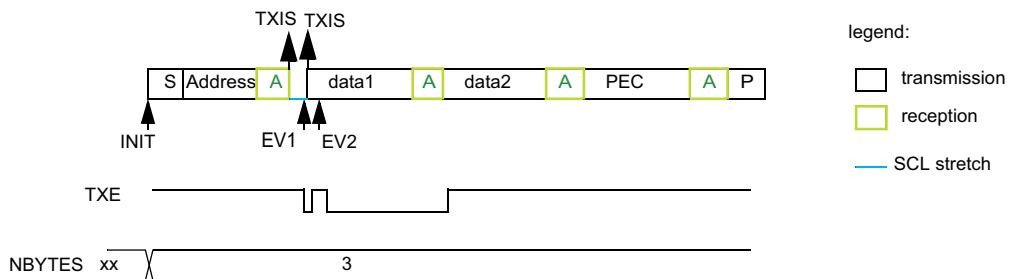
If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES - 1 have been transmitted, the I2C\_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 318. Bus transfer diagrams for SMBus master transmitter**

Example SMBus master transmitter 2 bytes + PEC, automatic end mode (STOP)

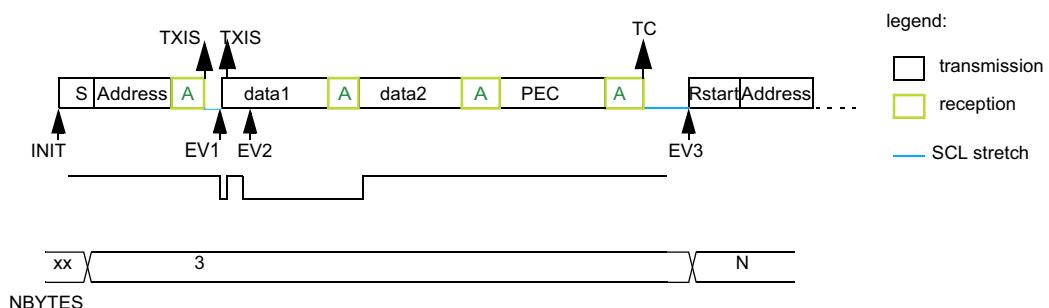


INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

Example SMBus master transmitter 2 bytes + PEC, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19871V2

### SMBus master receiver

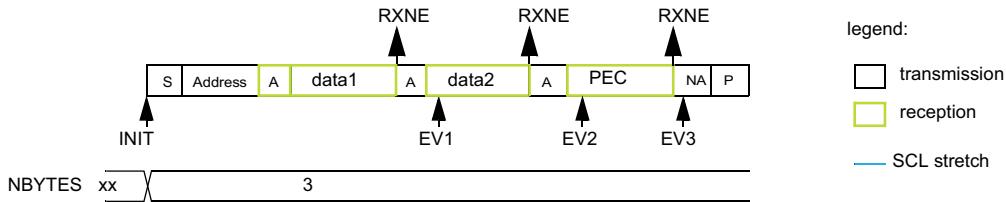
When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES - 1 data have been received, the next received byte is automatically checked versus the I2C\_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES - 1 data have been received, the next received byte is automatically checked versus the I2C\_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 319. Bus transfer diagrams for SMBus master receiver

Example SMBus master receiver 2 bytes + PEC, automatic end mode (STOP)



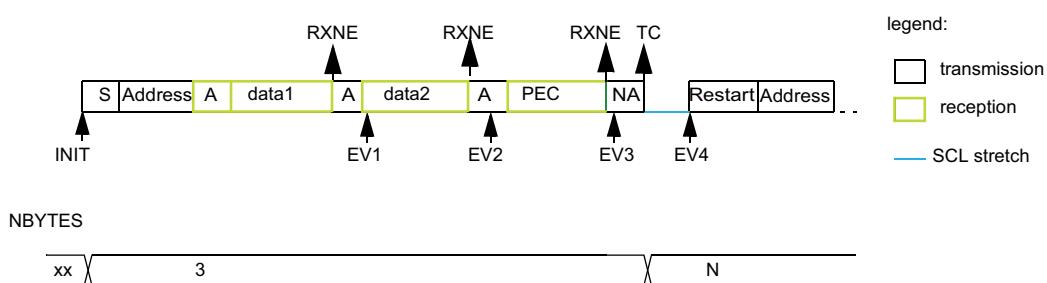
INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START

EV1: RXNE ISR: rd data1

EV2: RXNE ISR: rd data2

EV3: RXNE ISR: rd PEC

Example SMBus master receiver 2 bytes + PEC, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START

EV1: RXNE ISR: rd data1

EV2: RXNE ISR: rd data2

EV3: RXNE ISR: read PEC

EV4: TC ISR: program Slave address, program NBYTES = N, set START

MS19872V2

### 32.4.15 Wake-up from Stop mode on address match

This section is relevant only when wake-up from Stop mode feature is supported. Refer to [Section 32.3: I2C implementation](#).

The I2C is able to wake-up the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wake-up from Stop mode is enabled by setting the WUPEN bit in the I2C\_CR1 register. The HSI oscillator must be selected as the clock source for I2CCLK in order to allow wake-up from Stop mode.

During Stop mode, the HSI is switched off. When a START is detected, the I2C interface switches the HSI on, and stretches SCL low until HSI is woken up.

HSI is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wake-up time. The stretch is released when ADDR flag is cleared by software, and the transfer goes on normally.

If the address does not match, the HSI is switched off again and the MCU is not woken up.

**Note:** *If the I2C clock is the system clock, or if WUPEN = 0, the HSI is not switched on after a START is received.*

*Only an ADDR interrupt can wake-up the MCU. Therefore do not enter Stop mode when the I2C is performing a transfer as a master, or as an addressed slave after the ADDR flag is set. This can be managed by clearing SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.*

**Caution:** The digital filter is not compatible with the wake-up from Stop mode feature. If the DNF bit is not equal to 0, setting the WUPEN bit has no effect.

**Caution:** This feature is available only when the I2C clock source is the HSI oscillator.

**Caution:** Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

**Caution:** If wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

### 32.4.16 Error conditions

The following errors are the conditions which may cause communication to fail.

#### Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

#### Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Overrun/underrun error (OVR)

An overrun or underrun error is detected in slave mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF=1 and the first data byte must be sent. The content of the I2C\_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C\_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Packet error checking error (PECERR)

This section is relevant only when the SMBus feature is supported. Refer to [Section 32.3: I2C implementation](#).

A PEC error is detected when the received PEC byte does not match with the I2C\_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Timeout error (TIMEOUT)

This section is relevant only when the SMBus feature is supported. Refer to [Section 32.3: I2C implementation](#).

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE = 1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus  $t_{LOW:MEXT}$  parameter)
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus  $t_{LOW:SEXT}$  parameter)

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Alert (ALERT)

This section is relevant only when the SMBus feature is supported. Refer to [Section 32.3: I2C implementation](#).

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN = 1), the alert pin detection is enabled (ALERTEN = 1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### 32.4.17 DMA requests

#### Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit in the I2C\_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see ) to the I2C\_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to [Master transmitter](#).
- In slave mode:
  - With NOSTRETCH = 0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH = 1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to [SMBus slave transmitter](#) and [SMBus master transmitter](#).

*Note:* If DMA is used for transmission, the TXIE bit does not need to be enabled.

#### Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit in the I2C\_CR1 register. Data is loaded from the I2C\_RXDR register to an SRAM area configured using the DMA peripheral (refer to ) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In Master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.
- In Slave mode with NOSTRETCH = 0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see [Section 32.3: I2C implementation](#)): the PEC transfer is managed with the NBYTES counter. Refer to [SMBus Slave receiver](#) and [SMBus master receiver](#).

*Note:* If DMA is used for reception, the RXIE bit does not need to be enabled.

### 32.4.18 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the `DBG_I2Cx_` configuration bits in the DBG module.

## 32.5 I2C low-power modes

Table 201. Effect of low-power modes on the I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop <sup>(1)</sup>	The I2C registers content is kept. If <code>WUPEN</code> = 1 and I2C is clocked by an internal oscillator (HSI): the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. If <code>WUPEN</code> = 0: the I2C must be disabled before entering Stop mode
Standby	The I2C peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to [Section 32.3](#) for information about the Stop modes supported by each instance. If wake-up from a specific Stop mode is not supported, the instance must be disabled before entering this Stop mode.

## 32.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

Table 202. I2C Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
I2C	I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	No
		Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register		
		Stop detection interrupt flag	STOPF	STOPIE	Write STOPCF=1		
		Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] ≠ 0		
		Transfer complete	TC		Write START=1 or STOP=1		
		Address matched	ADDR	ADDRIE	Write ADDRCF=1		Yes <sup>(1)</sup>
		NACK reception	NACKF	NACKIE	Write NACKCF=1		No
I2C	I2C_ER	Bus error	BERR	ERRIE	Write BERRCF=1	Yes	No
		Arbitration loss	ARLO		Write ARLOCF=1		
		Overrun/Underrun	OVR		Write OVRCF=1		
		PEC error	PECERR		Write PECERRCF=1		
		Timeout/t <sub>LOW</sub> error	TIMEOUT		Write TIMEOUTCF=1		
		SMBus alert	ALERT		Write ALERTCF=1		

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the Wake-up from Stop mode feature. Refer to [Section 32.3: I2C implementation](#).

## 32.7 I2C registers

Refer to [Section 1.2 on page 61](#) for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

### 32.7.1 I2C control register 1 (I2C\_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to  $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$ .

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERT EN	SMBD EN	SMBH EN	GCEN	WUPE N	NOSTR ETCH	SBC
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMA EN	TXDMA EN	Res.	ANF OFF	DNF[3:0]				ERRIE	TCIE	STOP IE	NACK IE	ADDR IE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **PECEN**: PEC enable

- 0: PEC calculation disabled
- 1: PEC calculation enabled

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 32.3: I2C implementation](#).*

Bit 22 **ALERTEN**: SMBus alert enable

- 0: The SMBus alert pin (SMBA) is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN=0), the SMBA pin is released and the Alert Response Address header is disabled (0001100x followed by NACK).
- 1: The SMBus alert pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN=0), the SMBA pin is driven low and the Alert Response Address header is enabled (0001100x followed by ACK).

*Note: When ALERTEN=0, the SMBA pin can be used as a standard GPIO.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 32.3: I2C implementation](#).*

Bit 21 **SMBDEN**: SMBus device default address enable

- 0: Device default address disabled. Address 0b1100001x is NACKed.
- 1: Device default address enabled. Address 0b1100001x is ACKed.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 32.3: I2C implementation](#).*

Bit 20 **SMBHEN**: SMBus host address enable

- 0: Host address disabled. Address 0b0001000x is NACKed.
- 1: Host address enabled. Address 0b0001000x is ACKed.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 32.3: I2C implementation](#).*

Bit 19 **GCEN**: General call enable

- 0: General call disabled. Address 0b00000000 is NACKed.
- 1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable

- 0: Wake-up from Stop mode disable.
- 1: Wake-up from Stop mode enable.

*Note: If the Wake-up from Stop mode feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.*

*Note: WUPEN can be set only when DNF = '0000'*

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.

- 0: Clock stretching enabled
- 1: Clock stretching disabled

*Note: This bit can only be programmed when the I2C is disabled (PE = 0).*

Bit 16 **SBC**: Slave byte control

This bit is used to enable hardware byte control in slave mode.

- 0: Slave byte control disabled
- 1: Slave byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable

- 0: DMA mode disabled for reception
- 1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable

- 0: DMA mode disabled for transmission
- 1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF**: Analog noise filter OFF

- 0: Analog noise filter enabled
- 1: Analog noise filter disabled

*Note: This bit can only be programmed when the I2C is disabled (PE = 0).*

Bits 11:8 **DNF[3:0]**: Digital noise filter

These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to  $DNF[3:0] * t_{I2CCLK}$

0000: Digital filter disabled

0001: Digital filter enabled and filtering capability up to 1  $t_{I2CCLK}$

..  
1111: digital filter enabled and filtering capability up to 15  $t_{I2CCLK}$

*Note: If the analog filter is also enabled, the digital filter is added to the analog filter.*

*This filter can only be programmed when the I2C is disabled (PE = 0).*

Bit 7 **ERRIE**: Error interrupts enable

- 0: Error detection interrupts disabled
- 1: Error detection interrupts enabled

*Note: Any of these errors generate an interrupt:*

- Arbitration Loss (ARLO)*
- Bus Error detection (BERR)*
- Overrun/Underrun (OVR)*
- Timeout detection (TIMEOUT)*
- PEC error detection (PECERR)*
- Alert pin event detection (ALERT)*

Bit 6 **TCIE**: Transfer complete interrupt enable

- 0: Transfer complete interrupt disabled
- 1: Transfer complete interrupt enabled

*Note: Any of these events generate an interrupt:*

- Transfer complete (TC)*
- Transfer complete reload (TCR)*

Bit 5 **STOPIE**: Stop detection Interrupt enable

- 0: Stop detection (STOPF) interrupt disabled
- 1: Stop detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received Interrupt enable

- 0: Not acknowledge (NACKF) received interrupts disabled
- 1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match Interrupt enable (slave only)

- 0: Address match (ADDR) interrupts disabled
- 1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX Interrupt enable

- 0: Receive (RXNE) interrupt disabled
- 1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX Interrupt enable

- 0: Transmit (TXIS) interrupt disabled
- 1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable

- 0: Peripheral disable
- 1: Peripheral enable

*Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.*

### 32.7.2 I2C control register 2 (I2C\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to  $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$ .

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTO END	RE LOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD1 OR	ADD10	RD_ WRN	SADD[9:0]							rw	rw	rw
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.

0: No PEC transfer.

1: PEC transmission/reception is requested

*Note: Writing '0' to this bit has no effect.*

*This bit has no effect when RELOAD is set.*

*This bit has no effect in slave mode when SBC = 0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'.*

*Refer to Section 32.3: I2C implementation.*

Bit 25 **AUTOEND**: Automatic end mode (master mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

*Note: This bit has no effect in slave mode or when the RELOAD bit is set.*

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC = 0.

*Note: Changing these bits when the START bit is set is not allowed.*

Bit 15 **NACK**: NACK generation (slave mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

- 0: an ACK is sent after current received byte.
- 1: a NACK is sent after current received byte.

*Note: Writing '0' to this bit has no effect.*

*This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.*

*When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.*

*When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.*

Bit 14 **STOP**: Stop generation (master mode)

The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.

**In Master mode:**

- 0: No Stop generation.
- 1: Stop generation after current byte transfer.

*Note: Writing '0' to this bit has no effect.*

Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing '1' to the ADDRCF bit in the I2C\_ICR register.

- 0: No Start generation.
- 1: Restart/Start generation:

If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated start condition when RELOAD=0, after the end of the NBYTES transfer.

Otherwise setting this bit generates a START condition once the bus is free.

*Note: Writing '0' to this bit has no effect.*

*The START bit can be set even if the bus is BUSY or I2C is in slave mode.*

*This bit has no effect when RELOAD is set.*

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)

0: The master sends the complete 10 bit slave address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.

- 1: The master only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

*Note: Changing this bit when the START bit is set is not allowed.*

Bit 11 **ADD10**: 10-bit addressing mode (master mode)

- 0: The master operates in 7-bit addressing mode,
- 1: The master operates in 10-bit addressing mode

*Note: Changing this bit when the START bit is set is not allowed.*

Bit 10 **RD\_WRN**: Transfer direction (master mode)

- 0: Master requests a write transfer.
- 1: Master requests a read transfer.

*Note: Changing this bit when the START bit is set is not allowed.*

Bits 9:0 **SADD[9:0]**: Slave address (master mode)

**In 7-bit addressing mode (ADD10 = 0):**

SADD[7:1] must be written with the 7-bit slave address to be sent. The bits SADD[9], SADD[8] and SADD[0] are don't care.

**In 10-bit addressing mode (ADD10 = 1):**

SADD[9:0] must be written with the 10-bit slave address to be sent.

*Note: Changing these bits when the START bit is set is not allowed.*

### 32.7.3 I2C own address 1 register (I2C\_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1 MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 15 **OA1EN**: Own address 1 enable

0: Own address 1 disabled. The received slave address OA1 is NACKed.

1: Own address 1 enabled. The received slave address OA1 is ACKed.

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

Bit 10 **OA1MODE**: Own address 1 10-bit mode

0: Own address 1 is a 7-bit address.

1: Own address 1 is a 10-bit address.

*Note: This bit can be written only when OA1EN = 0.*

Bits 9:0 **OA1[9:0]**: Interface own slave address

7-bit addressing mode: OA1[7:1] contains the 7-bit own slave address. The bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address.

*Note: These bits can be written only when OA1EN = 0.*

### 32.7.4 I2C own address 2 register (I2C\_OAR2)

Address offset: 0x00C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 15 **OA2EN**: Own address 2 enable

- 0: Own address 2 disabled. The received slave address OA2 is NACKed.
- 1: Own address 2 enabled. The received slave address OA2 is ACKed.

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

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks

- 000: No mask
- 001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
- 010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
- 011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
- 100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
- 101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
- 110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
- 111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

*Note: These bits can be written only when OA2EN = 0.*

*As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.*

Bits 7:1 **OA2[7:1]**: Interface address

7-bit addressing mode: 7-bit address

*Note: These bits can be written only when OA2EN = 0.*

Bit 0 Reserved, must be kept at reset value.

### 32.7.5 I2C timing register (I2C\_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait states

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

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale I2CCLK in order to generate the clock period  $t_{PRESC}$  used for data setup and hold counters (refer to [I2C timings](#)) and for SCL high and low level counters (refer to [I2C master initialization](#)).

$$t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$$

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay  $t_{SCLDEL}$  between SDA edge and SCL rising edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during  $t_{SCLDEL}$ .

$$t_{SCLDEL} = (SCLDEL+1) \times t_{PRESC}$$

*Note:  $t_{SCLDEL}$  is used to generate  $t_{SU:DAT}$  timing.*

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay  $t_{SDADEL}$  between SCL falling edge and SDA edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during  $t_{SDADEL}$ .

$$t_{SDADEL} = SDADEL \times t_{PRESC}$$

*Note:  $t_{SDADEL}$  is used to generate  $t_{HD:DAT}$  timing.*

Bits 15:8 **SCLH[7:0]**: SCL high period (master mode)

This field is used to generate the SCL high period in master mode.

$$t_{SCLH} = (SCLH+1) \times t_{PRESC}$$

*Note:  $t_{SCLH}$  is also used to generate  $t_{SU:STO}$  and  $t_{HD:STA}$  timing.*

Bits 7:0 **SCLL[7:0]**: SCL low period (master mode)

This field is used to generate the SCL low period in master mode.

$$t_{SCLL} = (SCLL+1) \times t_{PRESC}$$

*Note:  $t_{SCLL}$  is also used to generate  $t_{BUF}$  and  $t_{SU:STA}$  timings.*

*Note: This register must be configured when the I2C is disabled (PE = 0).*

*Note: The STM32CubeMX tool calculates and provides the I2C\_TIMINGR content in the I2C Configuration window.*

### 32.7.6 I2C timeout register (I2C\_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to  $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$ .

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

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than  $t_{\text{LOW:EXT}}$  is done by the I2C interface, a timeout error is detected (TIMEOUT=1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

In master mode, the master cumulative clock low extend time ( $t_{\text{LOW:MEXT}}$ ) is detected

In slave mode, the slave cumulative clock low extend time ( $t_{\text{LOW:SEXT}}$ ) is detected

$t_{\text{LOW:EXT}} = (\text{TIMEOUTB}+1) \times 2048 \times \text{I2CCLK}$

*Note: These bits can be written only when TEXTEN=0.*

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled: when SCL is low for more than  $t_{\text{TIMEOUT}}$  (TIDLE=0) or high for more than  $t_{\text{IDLE}}$  (TIDLE=1), a timeout error is detected (TIMEOUT=1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

*Note: This bit can be written only when TIMOUTEN=0.*

Bits 11:0 **TIMEOUTA[11:0]**: Bus Timeout A

This field is used to configure:

The SCL low timeout condition  $t_{\text{TIMEOUT}}$  when TIDLE=0

$t_{\text{TIMEOUT}} = (\text{TIMEOUTA}+1) \times 2048 \times \text{I2CCLK}$

The bus idle condition (both SCL and SDA high) when TIDLE=1

$t_{\text{IDLE}} = (\text{TIMEOUTA}+1) \times 4 \times \text{I2CCLK}$

*Note: These bits can be written only when TIMOUTEN=0.*

**Note:** If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to [Section 32.3: I2C implementation](#).

### 32.7.7 I2C interrupt and status register (I2C\_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16							
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]														DIR
								r	r	r	r	r	r	r	r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
BUSY	Res.	ALERT	TIME OUT	PEC ERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE							
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs							

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 ADDCODE[6:0]: Address match code (Slave mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1).

In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the 2 MSBs of the address.

Bit 16 DIR: Transfer direction (Slave mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, slave enters receiver mode.

1: Read transfer, slave enters transmitter mode.

Bit 15 BUSY: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 ALERT: SMBus alert

This flag is set by hardware when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'.*

*Refer to Section 32.3: I2C implementation.*

Bit 12 TIMEOUT: Timeout or  $t_{LOW}$  detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'.*

*Refer to Section 32.3: I2C implementation.*

Bit 11 **PECERR**: PEC Error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 32.3: I2C implementation.*

Bit 10 **OVR**: Overrun/Underrun (slave mode)

This flag is set by hardware in slave mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced Start or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting the BERRCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 7 **TCR**: Transfer Complete Reload

This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

*Note: This bit is cleared by hardware when PE = 0.*

*This flag is only for master mode, or for slave mode when the SBC bit is set.*

Bit 6 **TC**: Transfer Complete (master mode)

This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 5 **STOPF**: Stop detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- either as a master, provided that the STOP condition is generated by the peripheral.
- or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 4 **NACKF**: Not Acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 3 **ADDR**: Address matched (slave mode)

This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting the ADDRRCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C\_RXDR register, and is ready to be read. It is cleared when I2C\_RXDR is read.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C\_TXDR register is empty and the data to be transmitted must be written in the I2C\_TXDR register. It is cleared when the next data to be sent is written in the I2C\_TXDR register.

This bit can be written to '1' by software when NOSTRETCH = 1 only, in order to generate a TXIS event (interrupt if TXIE=1 or DMA request if TXDMAEN = 1).

*Note: This bit is cleared by hardware when PE = 0.*

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C\_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C\_TXDR register.

This bit can be written to '1' by software in order to flush the transmit data register I2C\_TXDR.

*Note: This bit is set by hardware when PE = 0.*

### 32.7.8 I2C interrupt clear register (I2C\_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERT CF	TIMOU TCF	PECCF	OVRCF	ARLOC F	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDR CF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

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

Bit 13 **ALERTCF**: Alert flag clear

Writing 1 to this bit clears the ALERT flag in the I2C\_ISR register.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 32.3: I2C implementation](#).*

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C\_ISR register.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 32.3: I2C implementation](#).*

Bit 11 **PECCF**: PEC Error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C\_ISR register.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 32.3: I2C implementation](#).*

- Bit 10 **OVRCF**: Overrun/Underrun flag clear  
Writing 1 to this bit clears the OVR flag in the I2C\_ISR register.
- Bit 9 **ARLOCF**: Arbitration lost flag clear  
Writing 1 to this bit clears the ARLO flag in the I2C\_ISR register.
- Bit 8 **BERRCF**: Bus error flag clear  
Writing 1 to this bit clears the BERRF flag in the I2C\_ISR register.
- Bits 7:6 Reserved, must be kept at reset value.
- Bit 5 **STOPCF**: STOP detection flag clear  
Writing 1 to this bit clears the STOPF flag in the I2C\_ISR register.
- Bit 4 **NACKCF**: Not Acknowledge flag clear  
Writing 1 to this bit clears the NACKF flag in I2C\_ISR register.
- Bit 3 **ADDRCF**: Address matched flag clear  
Writing 1 to this bit clears the ADDR flag in the I2C\_ISR register. Writing 1 to this bit also clears the START bit in the I2C\_CR2 register.
- Bits 2:0 Reserved, must be kept at reset value.

### 32.7.9 I2C PEC register (I2C\_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: No wait states

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

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

Bits 7:0 **PEC[7:0]**: Packet error checking register

This field contains the internal PEC when PECEN=1.

The PEC is cleared by hardware when PE = 0.

**Note:** *If the SMBus feature is not supported, this register is reserved and forced by hardware to “0x00000000”. Refer to [Section 32.3: I2C implementation](#).*

### 32.7.10 I2C receive data register (I2C\_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: No wait states

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

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

Bits 7:0 **RXDATA[7:0]**: 8-bit receive data

Data byte received from the I<sup>2</sup>C bus

### 32.7.11 I2C transmit data register (I2C\_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDATA[7:0]															
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	rw							

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]**: 8-bit transmit data

Data byte to be transmitted to the I<sup>2</sup>C bus

*Note: These bits can be written only when TXE = 1.*

### 32.7.12 I2C register map

The table below provides the I2C register map and reset values.

**Table 203. I2C register map and reset values**

**Table 203. I2C register map and reset values (continued)**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	I2C_TXDR	Res	TXDATA[7:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 33 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

### 33.1 USART introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and Half-duplex Single-wire communications, as well as LIN (local interconnection network), Smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and Modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

### 33.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data
  - Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wakeup from Mute mode by idle line detection or address mark detection
- Wakeup from Stop mode

### 33.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for Smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

### 33.4 USART implementation

The table below describes USART implementation. It also includes LPUART for comparison.

Table 204. USART / LPUART features

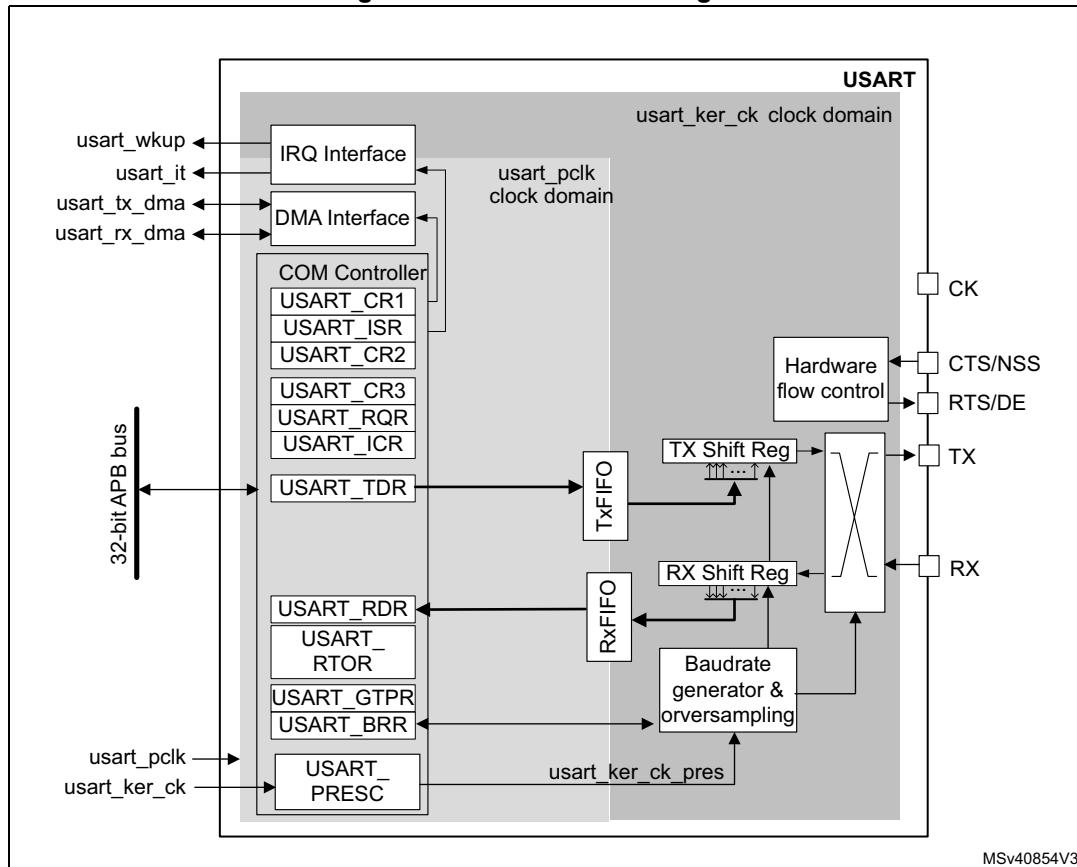
USART / LPUART modes/features <sup>(1)</sup>	USART1	LPUART1
Hardware flow control for modem	X	X
Continuous communication using DMA	X	X
Multiprocessor communication	X	X
Synchronous mode (Master/Slave)	X	-
Smartcard mode	X	-
Single-wire Half-duplex communication	X	X
IrDA SIR ENDEC block	X	-
LIN mode	X	-
Dual clock domain and wakeup from low-power mode	X	X
Receiver timeout interrupt	X	-
Modbus communication	X	-
Auto baud rate detection	X	-
Driver Enable	X	X
USART data length	7, 8 and 9 bits	
Tx/Rx FIFO	X	X
Tx/Rx FIFO size	8	

1. X = supported.

## 33.5 USART functional description

### 33.5.1 USART block diagram

Figure 320. USART block diagram



MSv40854V3

The simplified block diagram given in [Figure 320](#) shows two fully-independent clock domains:

- The **usart\_pclk** clock domain
 

The **usart\_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
- The **usart\_ker\_ck** kernel clock domain.
 

The **usart\_ker\_ck** is the USART clock source. It is independent from **usart\_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart\_ker\_ck** clock is stopped.

When the dual clock domain feature is disabled, the **usart\_ker\_ck** clock is the same as the **usart\_pclk** clock.

There is no constraint between **usart\_pclk** and **usart\_ker\_ck**: **usart\_ker\_ck** can be faster or slower than **usart\_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart\_ker\_ck** clock must be at least 3 times faster than the clock on the CK input.

### 33.5.2 USART signals

#### USART bidirectional communications

USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)

RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

- **TX** (Transmit Data Output)

When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In Single-wire and Smartcard modes, this I/O is used to transmit and receive data.

#### RS232 Hardware flow control mode

The following pins are required in RS232 Hardware flow control mode:

- **CTS** (Clear To Send)

When driven high, this signal blocks the data transmission at the end of the current transfer.

- **RTS** (Request To Send)

When it is low, this signal indicates that the USART is ready to receive data.

#### RS485 Hardware control mode

The following pin is required in RS485 Hardware control mode:

- **DE** (Driver Enable)

This signal activates the transmission mode of the external transceiver.

*Note:* DE and RTS share the same pin.

#### Synchronous master/slave mode and Smartcard mode

The following pin is required in synchronous master/slave mode and Smartcard mode:

- **CK**

This pin acts as Clock output in Synchronous master and Smartcard modes.

It acts as Clock input in Synchronous slave mode.

In Synchronous Master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.

In Smartcard mode, CK output provides the clock to the smartcard.

- **NSS**

This pin acts as Slave Select input in Synchronous slave mode.

*Note:* NSS and CTS share the same pin.

### 33.5.3 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART\_CR1 register (see [Figure 321](#)):

- 7-bit character length: M[1:0] = '10'
- 8-bit character length: M[1:0] = '00'
- 9-bit character length: M[1:0] = '01'

**Note:** *In 7-bit data length mode, the Smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.*

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

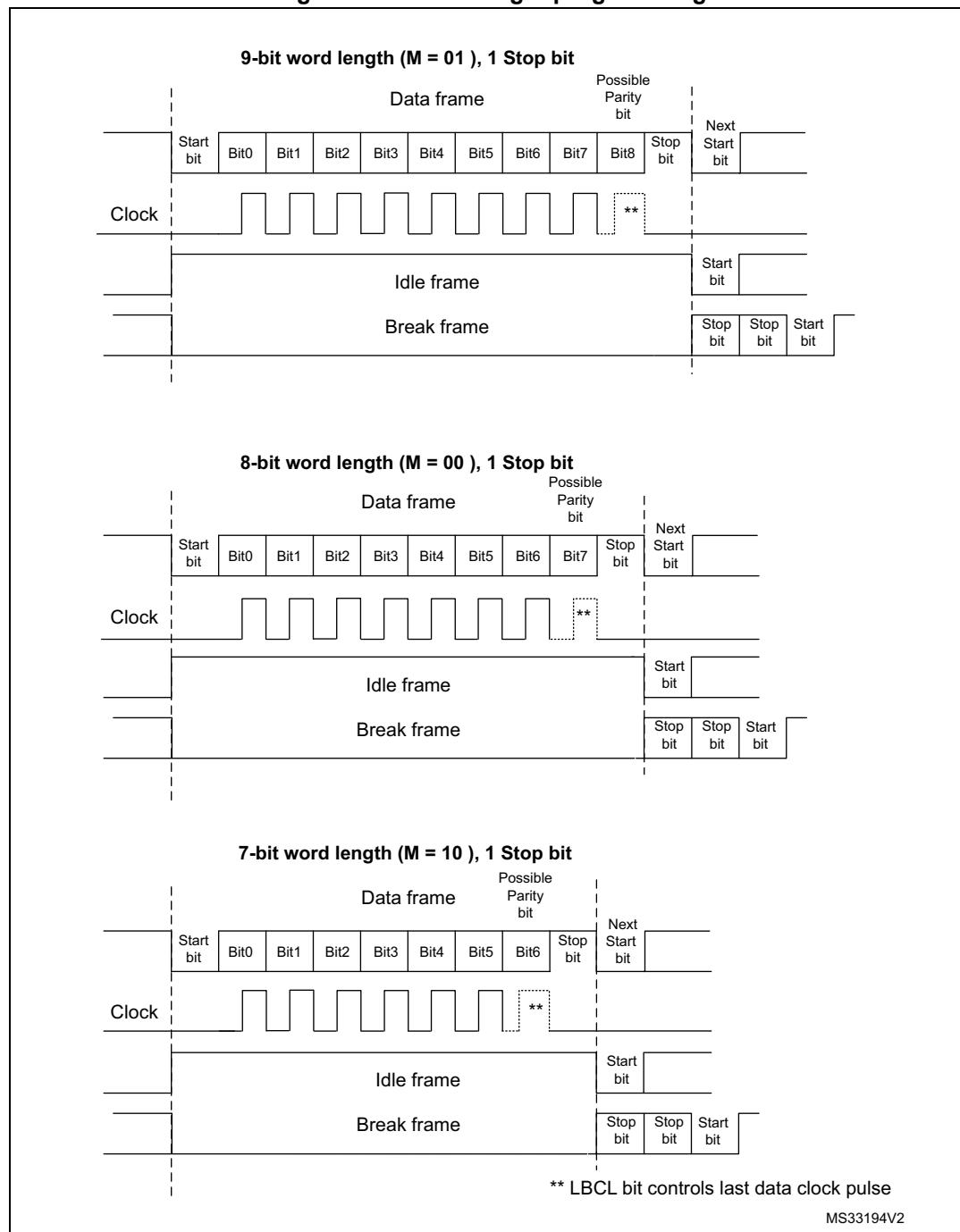
An **Idle character** is interpreted as an entire frame of "1"s (the number of "1"s includes the number of stop bits).

A **Break character** is interpreted on receiving "0"s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 321. Word length programming



### 33.5.4 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART\_CR1 register (bit 29). This mode is supported only in UART, SPI and Smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

*Note:* *The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.*

*The status flags are available in the USART\_ISR register.*

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART\_CR3 control register.

In this case:

- The RXFT flag is set in the USART\_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

RXFTCFG data have been received: one data in USART\_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to '101', the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the USART\_RDR). As a result, the next received data is not set the overrun flag.

- The TXFT flag is set in the USART\_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

### 33.5.5 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

#### Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART\_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART\_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

**Note:** *The TE bit must be set before writing the data to be transmitted to the USART\_TDR.*

*The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.*

*An idle frame is sent when the TE bit is enabled.*

#### Configurable stop bits

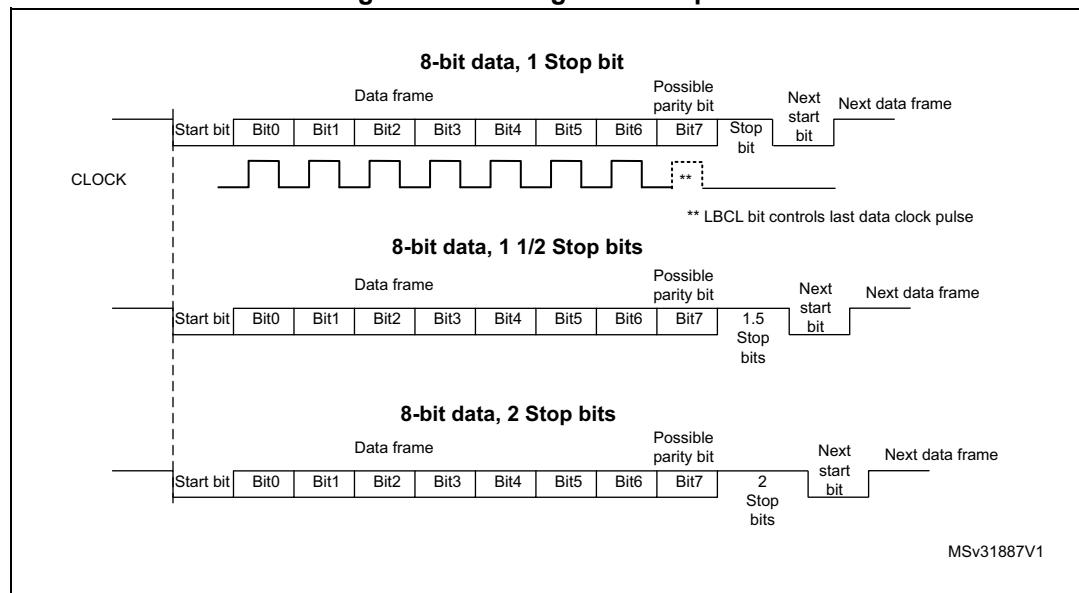
The number of stop bits to be transmitted with every character can be programmed in USART\_CR2, bits 13,12.

- **1 stop bit:** This is the default value of number of stop bits.
- **2 stop bits:** This is supported by normal USART, Single-wire and Modem modes.
- **1.5 stop bits:** To be used in Smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission features 10 low bits (when M[1:0] = '00') or 11 low bits (when M[1:0] = '01') or 9 low bits (when M[1:0] = '10') followed by 2 stop bits (see [Figure 322](#)). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 322. Configurable stop bits**



### Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART\_CR1 to define the word length.
2. Select the desired baud rate using the USART\_BRR register.
3. Program the number of stop bits in USART\_CR2.
4. Enable the USART by writing the UE bit in USART\_CR1 register to 1.
5. Select DMA enable (DMAT) in USART\_CR3 if multibuffer communication must take place. Configure the DMA register as explained in [Section 33.5.10: USART multiprocessor communication](#).
6. Set the TE bit in USART\_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART\_TDR register. Repeat this for each data to be transmitted in case of single buffer.
  - When FIFO mode is disabled, writing a data to the USART\_TDR clears the TXE flag.
  - When FIFO mode is enabled, writing a data to the USART\_TDR adds one data to the TXFIFO. Write operations to the USART\_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART\_TDR register, wait until TC = 1.
  - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
  - When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

## Single byte communication

- When FIFO mode is disabled

Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:

- the data have been moved from the USART\_TDR register to the shift register and the data transmission has started;
- the USART\_TDR register is empty;
- the next data can be written to the USART\_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is ongoing, a write instruction to the USART\_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

When no transmission is ongoing, a write instruction to the USART\_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:

- the TXFIFO is not full;
- the USART\_TDR register is empty;
- the next data can be written to the USART\_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART\_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

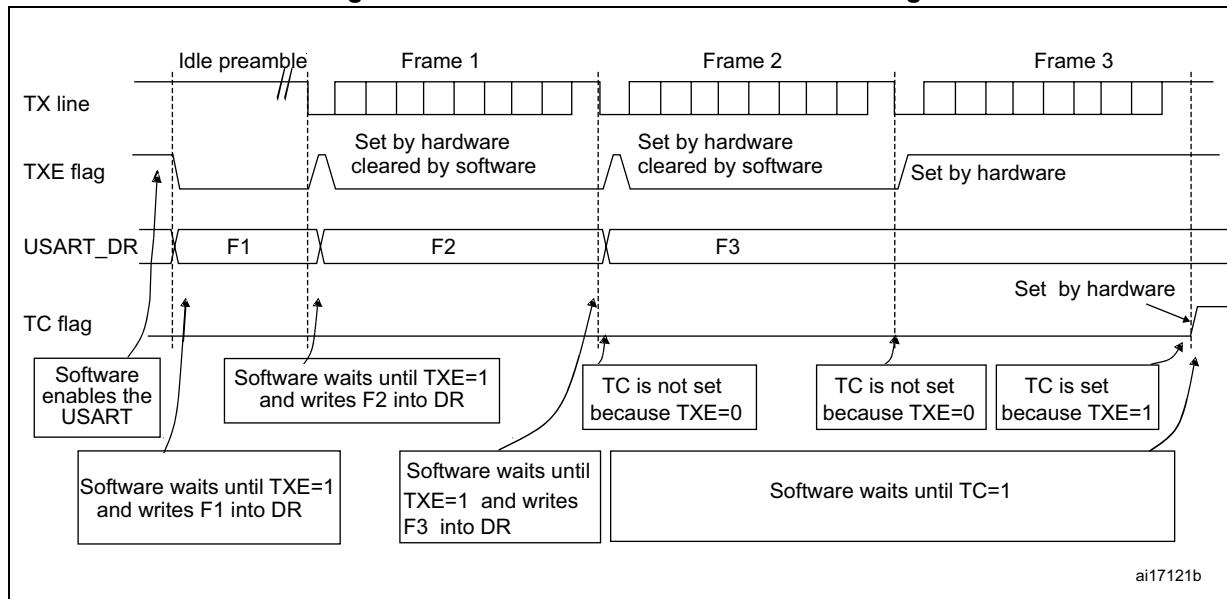
When the TXFIFO is not full, the TXFNF flag stays at '1' even after a write operation to USART\_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART\_CR1 register.

After writing the last data to the USART\_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the device to enter the low-power mode (see [Figure 323: TC/TXE behavior when transmitting](#)).

Figure 323. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

### Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see [Figure 321](#)).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

### Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

## 33.5.6 USART receiver

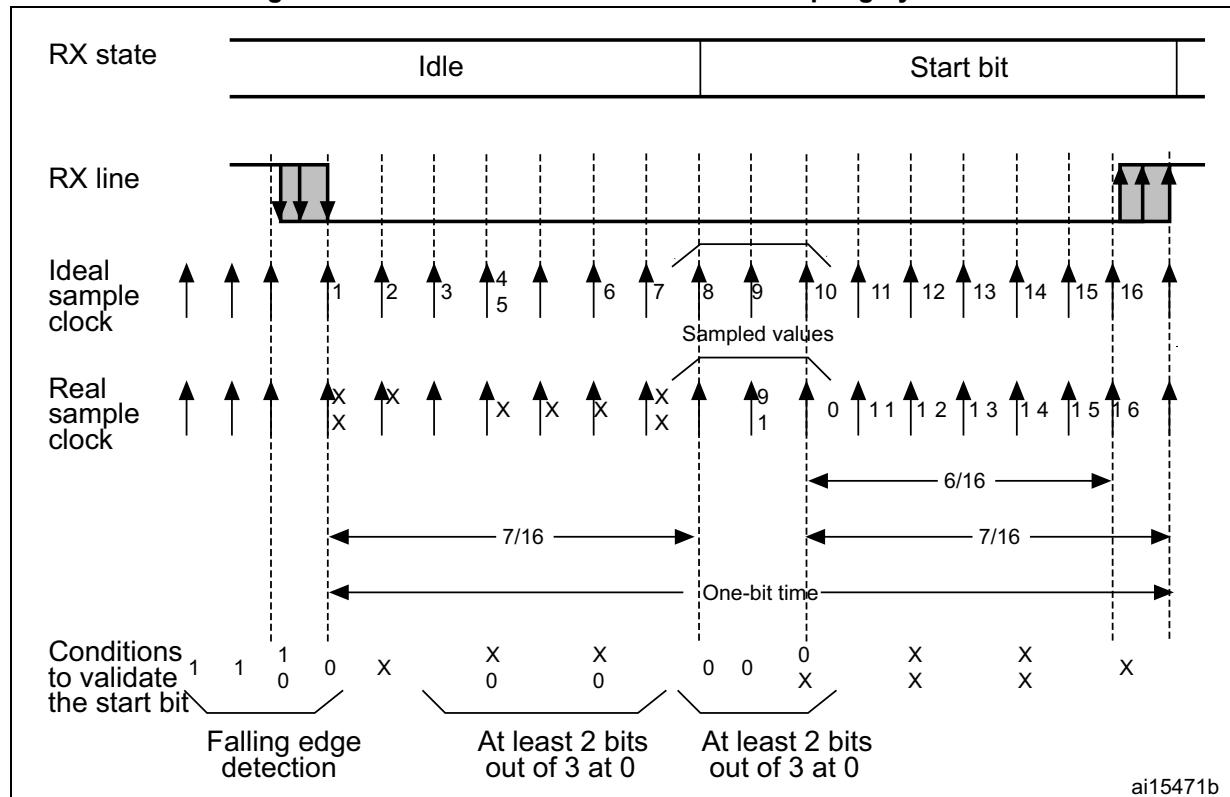
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART\_CR1 register.

### Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0 X 0 X 0 X 0 X 0.

Figure 324. Start bit detection when oversampling by 16 or 8



**Note:** If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at '0' (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at '0' and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at '0').

The start bit is validated but the NE noise flag is set if,

- a) for both samplings, 2 out of the 3 sampled bits are at '0' (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)
- or
- b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at '0'.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

## Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

### Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART\_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART\_BRR
3. Program the number of stop bits in USART\_CR2.
4. Enable the USART by writing the UE bit in USART\_CR1 register to '1'.
5. Select DMA enable (DMAR) in USART\_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in [Section 33.5.10: USART multiprocessor communication](#).
6. Set the RE bit USART\_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART\_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty i.e. when there are data to be read from the RXFIFO.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART\_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to '1' in the USART\_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART\_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to '1' in USART\_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

### Break character

When a break character is received, the USART handles it as a framing error.

### Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

### Overrun error

- FIFO mode disabled

An overrun error occurs if a character is received and RXNE has not been reset.

Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

- the ORE bit is set;
- the RDR content is not lost. The previous data is available by reading the USART\_RDR register.
- the shift register is overwritten. After that, any data received during overrun is lost.
- an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

Data can not be transferred from the shift register to the USART\_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

- The ORE bit is set.
- The first entry in the RXFIFO is not lost. It is available by reading the USART\_RDR register.
- The shift register is overwritten. After that point, any data received during overrun is lost.
- An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART\_ICR register.

*Note:*

*The ORE bit, when set, indicates that at least 1 data has been lost.*

*When the FIFO mode is disabled, there are two possibilities*

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

### Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see Section *Reset and clock control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

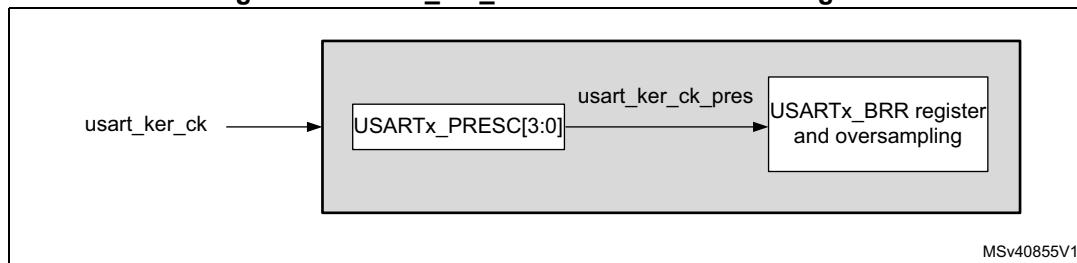
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is `usart_ker_ck`.

When the dual clock domain and the wakeup from low-power mode features are supported, the `usart_ker_ck` clock source can be configurable in the RCC (see Section *Reset and clock control (RCC)*). Otherwise the `usart_ker_ck` clock is the same as `usart_pclk`.

The `usart_ker_ck` clock can be divided by a programmable factor, defined in the `USART_PRESC` register.

**Figure 325. `usart_ker_ck` clock divider block diagram**



Some `usart_ker_ck` sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the `USART_RDR` register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the `OVER8` bit in the `USART_CR1` register either to 16 or 8 times the baud rate clock (see [Figure 326](#) and [Figure 327](#)).

Depending on your application:

- select oversampling by 8 (`OVER8 = 1`) to achieve higher speed (up to `usart_ker_ck_pres/8`). In this case the maximum receiver tolerance to clock deviation is reduced (refer to [Section 33.5.8: Tolerance of the USART receiver to clock deviation on page 1055](#))
- select oversampling by 16 (`OVER8 = 0`) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum

`uart_ker_ck_pres/16` (where `uart_ker_ck_pres` is the USART input clock divided by a prescaler).

Programming the `ONEBIT` bit in the `USART_CR3` register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the `NE` bit is set.
- A single sample in the center of the received bit

Depending on your application:

- select the three sample majority vote method (`ONEBIT = 0`) when operating in a noisy environment and reject the data when a noise is detected (refer to [Figure 205](#)) because this indicates that a glitch occurred during the sampling.
- select the single sample method (`ONEBIT = 1`) when the line is noise-free to increase the receiver tolerance to clock deviations (see [Section 33.5.8: Tolerance of the USART receiver to clock deviation on page 1055](#)). In this case the `NE` bit is never set.

When noise is detected in a frame:

- The `NE` bit is set at the rising edge of the `RXNE` bit (`RXFNE` in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the `USART_RDR` register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the `RXNE` bit (`RXFNE` in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the `EIE` bit is set in the `USART_CR3` register.

The `NE` bit is reset by setting `NECF` bit in `USART_ICR` register.

*Note:* *Noise error is not supported in SPI mode.*

*Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the `OVER8` bit is forced to '0' by hardware.*

**Figure 326. Data sampling when oversampling by 16**

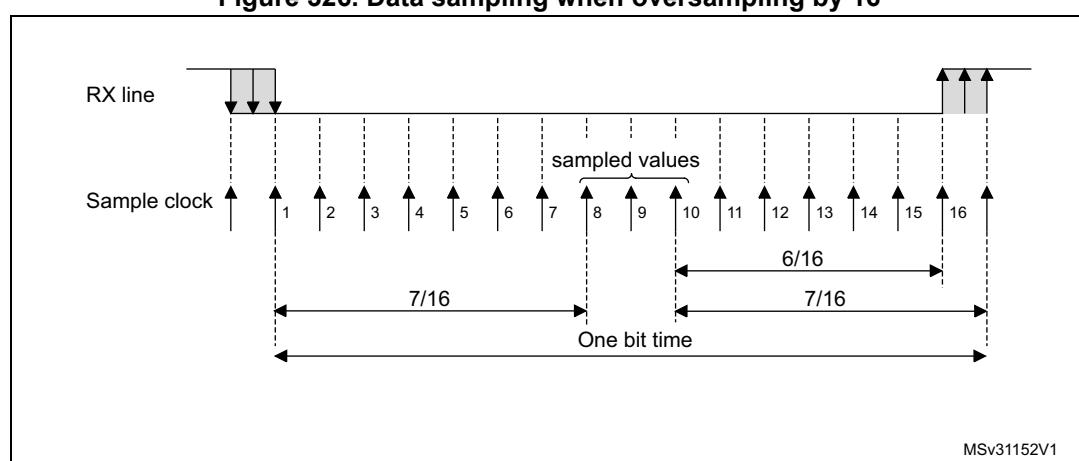


Figure 327. Data sampling when oversampling by 8

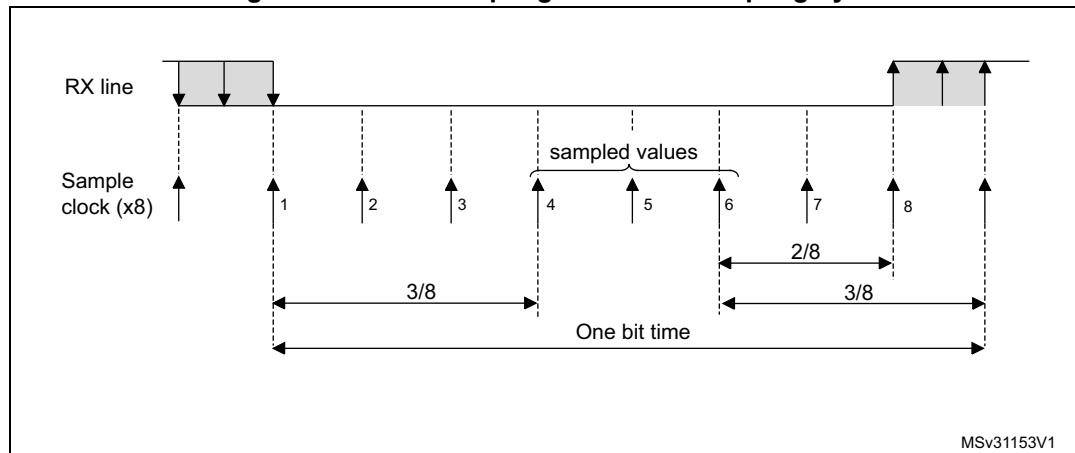


Table 205. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

### Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the USART\_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART\_CR3 register.

The FE bit is reset by writing '1' to the FECF in the USART\_ICR register.

*Note:* *Framing error is not supported in SPI mode.*

### Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART\_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

- **0.5 stop bit (reception in Smartcard mode):** no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit:** sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (Smartcard mode)**

When transmitting in Smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART\_CR1) and the stop bit is checked to test if the Smartcard has detected a parity error.

In the event of a parity error, the Smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to [Section 33.5.16: USART receiver timeout on page 1069](#) for more details).

- **2 stop bits**

Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

The framing error flag is set if a framing error is detected during the first stop bit.

The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

### 33.5.7 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART\_BRR register.

#### Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = '0' or '1')

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{USART\_ker\_ckpres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{USART\_ker\_ckpres}}{\text{USARTDIV}}$$

#### Equation 2: baud rate in Smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{USART\_ker\_ckpres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART\_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

**Note:** *The baud counters are updated to the new value in the baud registers after a write operation to USART\_BRR. Hence the baud rate register value should not be changed during communication.*

*In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.*

### How to derive USARTDIV from USART\_BRR register values

#### Example 1

To obtain 9600 baud with usart\_ker\_ck\_pres = 8 MHz:

- In case of oversampling by 16:  
USARTDIV = 8 000 000/9600  
BRR = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:  
USARTDIV = 2 \* 8 000 000/9600  
USARTDIV = 1666,66 (0d1667 = 0x683)  
BRR[3:0] = 0x3 >> 1 = 0x1  
BRR = 0x681

#### Example 2

To obtain 921.6 Kbaud with usart\_ker\_ck\_pres = 48 MHz:

- In case of oversampling by 16:  
USARTDIV = 48 000 000/921 600  
BRR = USARTDIV = 0d52 = 0x34
- In case of oversampling by 8:  
USARTDIV = 2 \* 48 000 000/921 600  
USARTDIV = 104 (0d104 = 0x68)  
BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4  
BRR = 0x64

### 33.5.8 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{USART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.

when  $M[1:0] = 01$ :

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when  $M[1:0] = 00$ :

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when  $M[1:0] = 10$ :

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$  is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in [Table 206](#), [Table 207](#), depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART\_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART\_CR1 register
- Bits BRR[3:0] of USART\_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART\_CR3 register.

**Table 206. Tolerance of the USART receiver when BRR [3:0] = 0000**

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

**Table 207. Tolerance of the USART receiver when BRR[3:0] is different from 0000**

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

**Note:** The data specified in [Table 206](#) and [Table 207](#) may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9-bit times when M = 10).

### 33.5.9 USART auto baud rate detection

The USART can detect and automatically set the USART\_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart\_ker\_ck\_pres/65535 and usart\_ker\_ck\_pres/16.
- When oversampling by 8, the baud rate ranges from usart\_ker\_ck\_pres/65535 and usart\_ker\_ck\_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART\_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0:** Any character starting with a bit at '1'.  
In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.  
In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2:** A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).  
In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3:** A 0x55 character frame.  
In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART\_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART\_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART\_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a '0').

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART\_ISR register.

**Note:** *The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.*

### 33.5.10 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the USART\_CR1 register.

**Note:** *When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart\_ker\_ck cycles), otherwise Mute mode might remain active.*

When the Mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART\_ISR register is set to '1'. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART\_RQR register, under certain conditions.

The USART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the USART\_CR1 register:

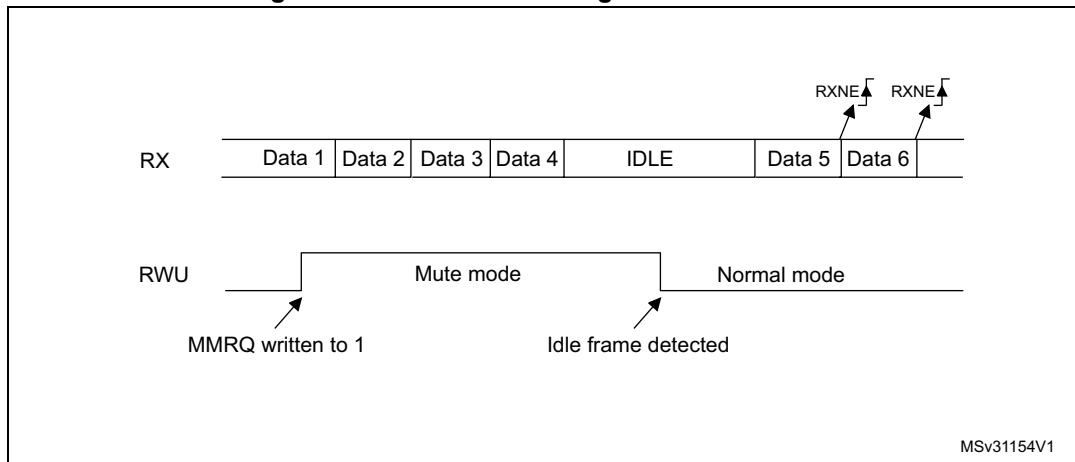
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

#### Idle line detection (WAKE = 0)

The USART enters Mute mode when the MMRQ bit is written to '1' and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART\_ISR register. An example of Mute mode behavior using Idle line detection is given in [Figure 328](#).

Figure 328. Mute mode using Idle line detection



**Note:** If the MMRQ is set while the IDLE character has already elapsed, Mute mode is not entered (RWU is not set).

If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

#### 4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a '1', otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART\_CR2 register.

**Note:** In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters Mute mode. When FIFO management is enabled, the software should ensure that there is at least one empty location in the RXFIFO before entering Mute mode.

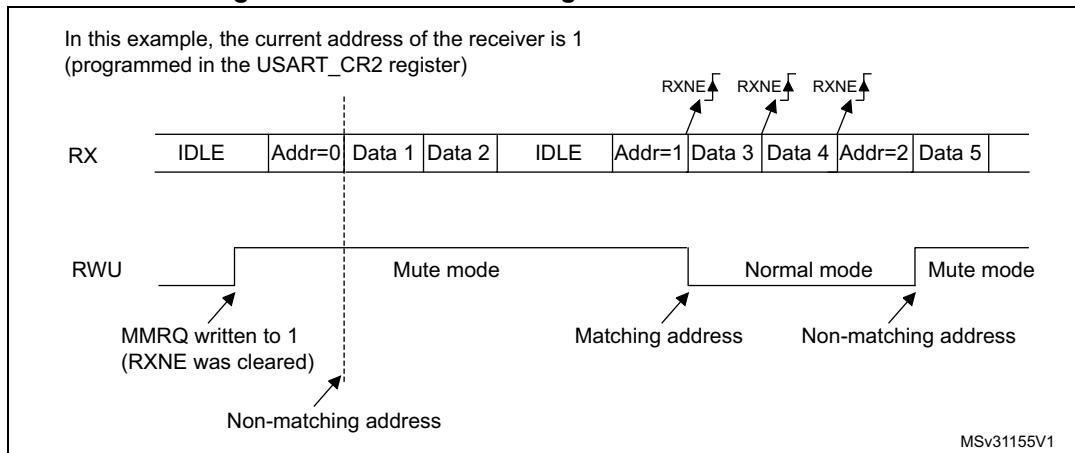
The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

**Note:** When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in Mute mode

An example of Mute mode behavior using address mark detection is given in [Figure 329](#).

Figure 329. Mute mode using address mark detection



### 33.5.11 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a Half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

#### Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART\_CR2 register and the RTOIE in the USART\_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

#### Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

### 33.5.12 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART\_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in [Table 208](#).

**Table 208. USART frame formats**

M bits	PCE bit	USART frame <sup>(1)</sup>
00	0	SB   8 bit data   STB
00	1	SB   7-bit data   PB   STB
01	0	SB   9-bit data   STB
01	1	SB   8-bit data PB   STB
10	0	SB   7bit data   STB
10	1	SB   6-bit data   PB   STB

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

#### Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits are set, the parity bit is equal to 0 if even parity is selected (PS bit in USART\_CR1 = 0).

#### Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART\_CR1 = 1).

#### Parity checking in reception

If the parity check fails, the PE flag is set in the USART\_ISR register and an interrupt is generated if PEIE is set in the USART\_CR1 register. The PE flag is cleared by software writing 1 to the PECE in the USART\_ICR register.

#### Parity generation in transmission

If the PCE bit is set in USART\_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS=1)).

### 33.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to [Section 33.4: USART implementation on page 1038](#).

The LIN mode is selected by setting the LINEN bit in the USART\_CR2 register. In LIN mode, the following bits must be kept cleared:

- CLKEN in the USART\_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USART\_CR3 register.

#### LIN transmission

The procedure described in [Section 33.5.4](#) has to be applied for LIN Master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 '0' bits as a break character. Then two bits of value '1' are sent to enable the next start detection.

#### LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART\_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART\_CR2) or 11 (when LBDL = 1 in USART\_CR2) consecutive bits are detected as '0', and are followed by a delimiter character, the LBDF flag is set in USART\_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a '1' is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

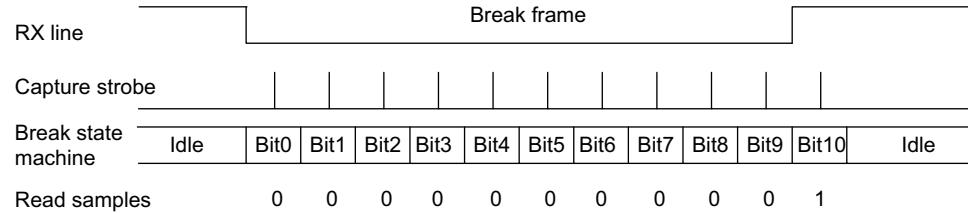
If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (i.e. stop bit detected at '0', which is the case for any break frame), the receiver stops until the break detection circuit receives either a '1', if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the [Figure 330: Break detection in LIN mode \(11-bit break length - LBDL bit is set\) on page 1064](#).

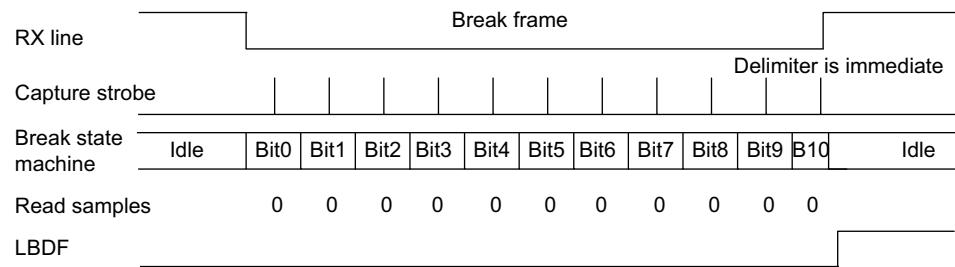
Examples of break frames are given on [Figure 331: Break detection in LIN mode vs. Framing error detection on page 1065](#).

Figure 330. Break detection in LIN mode (11-bit break length - LBDL bit is set)

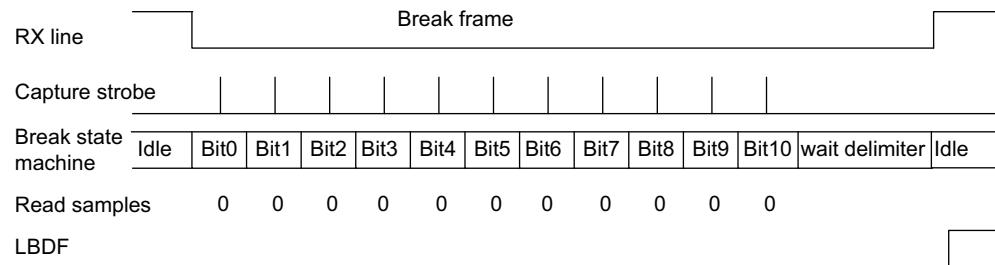
**Case 1: break signal not long enough => break discarded, LBDF is not set**



**Case 2: break signal just long enough => break detected, LBDF is set**

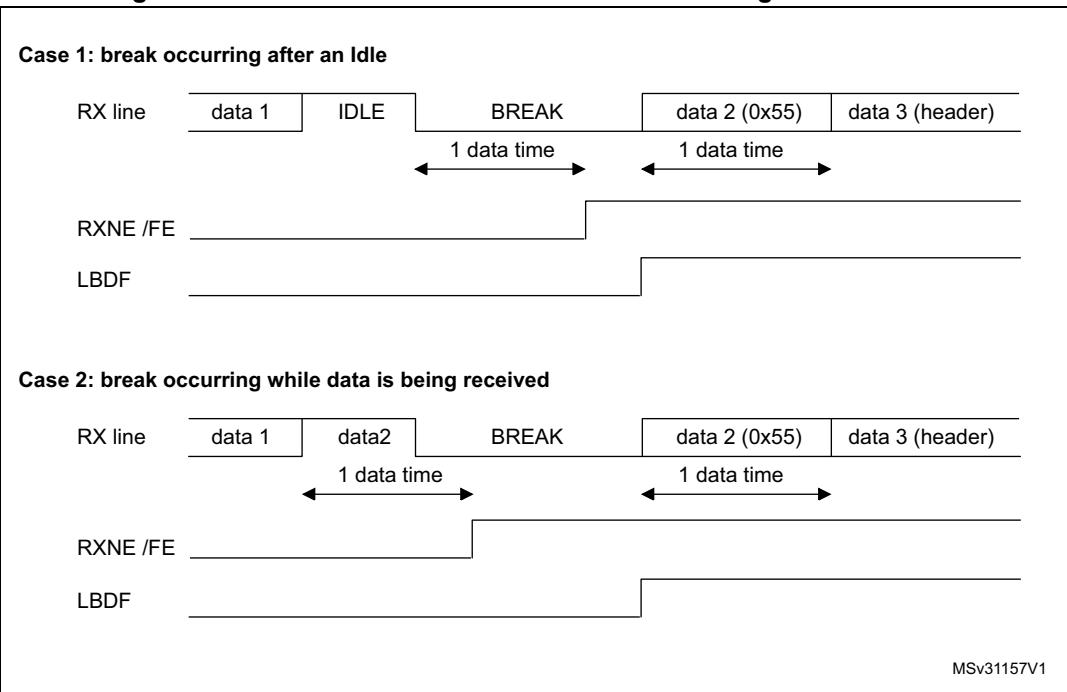


**Case 3: break signal long enough => break detected, LBDF is set**



MSv31156V1

Figure 331. Break detection in LIN mode vs. Framing error detection



### 33.5.14 USART synchronous mode

#### Master mode

The synchronous master mode is selected by programming the CLEN bit in the USART\_CR2 register to '1'. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART\_CR2 register,
- SCEN, HDSEL and IREN bits in the USART\_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART\_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART\_CR2 register is used to select the clock polarity, and the CPHA bit in the USART\_CR2 register is used to select the phase of the external clock (see [Figure 332](#), [Figure 333](#) and [Figure 334](#)).

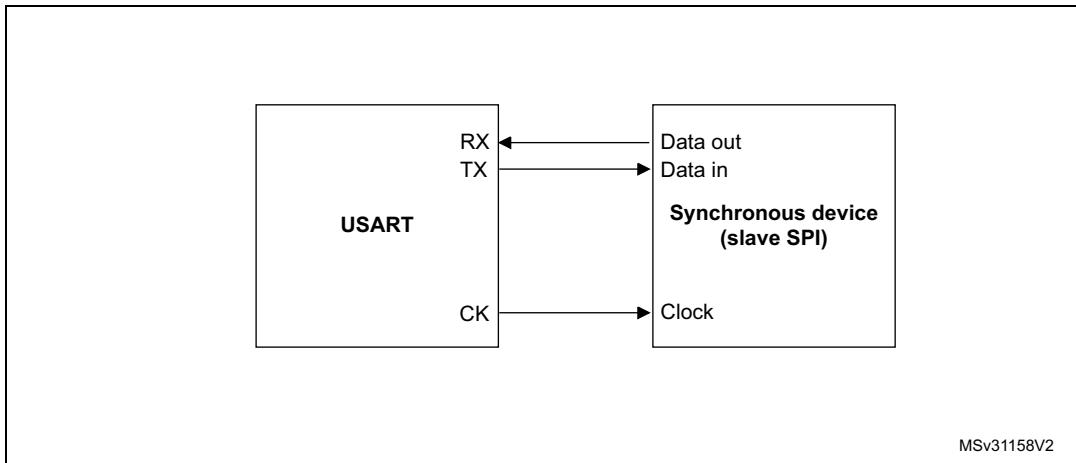
During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

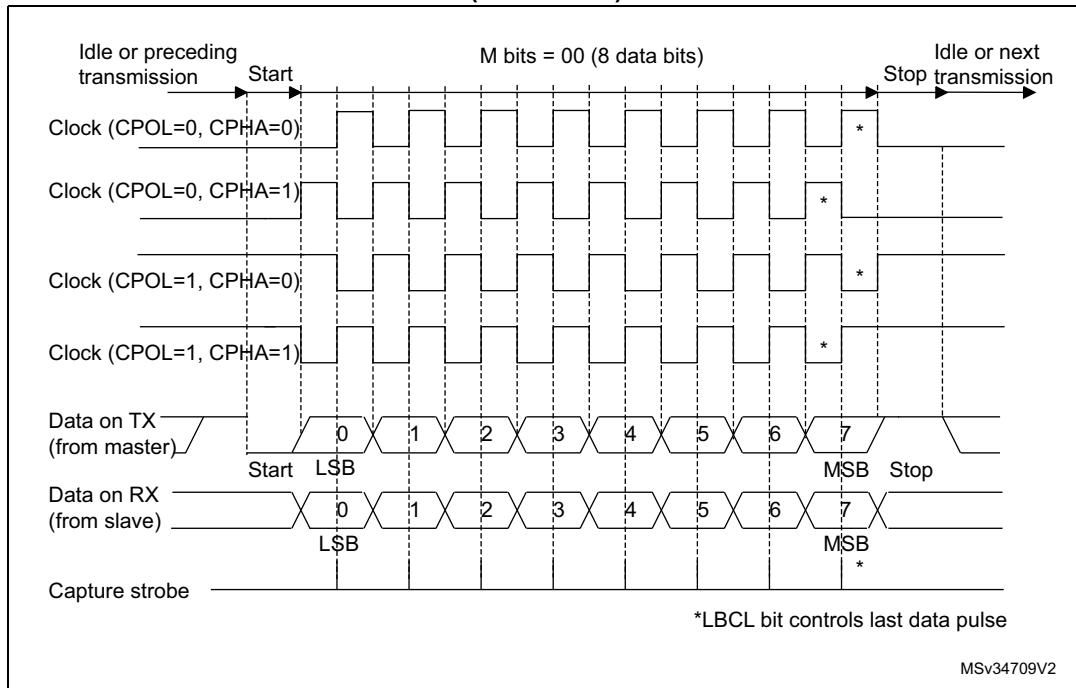
In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

**Note:** In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled ( $TE = 1$ ) and data are being transmitted (USART\_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

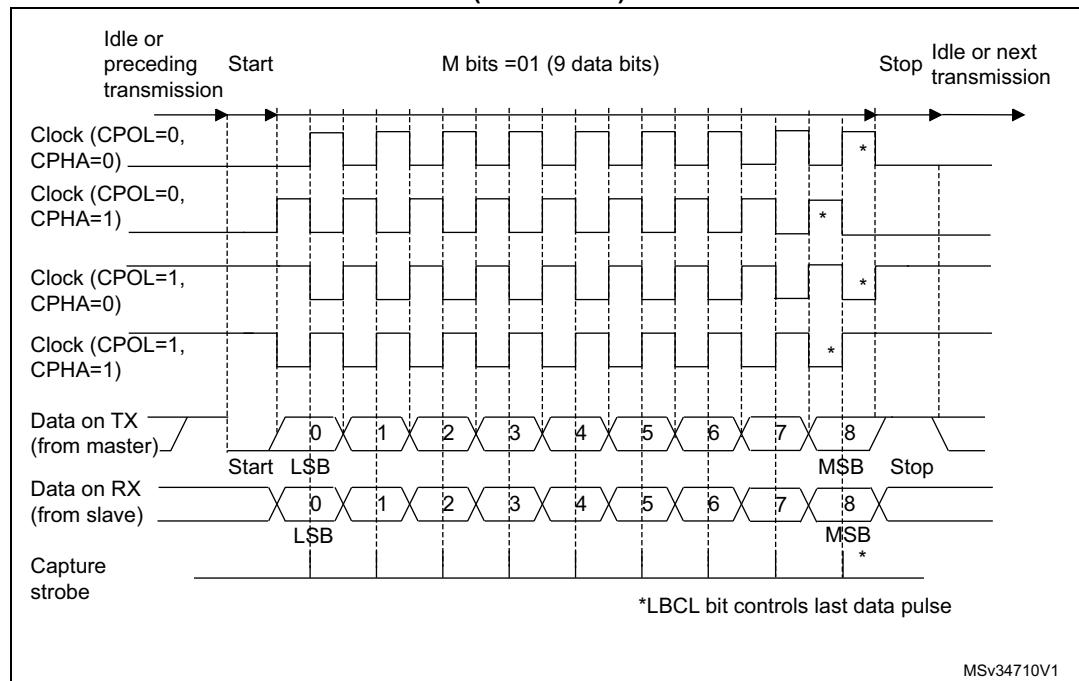
**Figure 332. USART example of synchronous master transmission**



**Figure 333. USART data clock timing diagram in synchronous master mode (M bits = 00)**



**Figure 334. USART data clock timing diagram in synchronous master mode  
(M bits = 01)**



### Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART\_CR2 register to '1'. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART\_CR2 register,
- SCEN, HDSEL and IREN bits in the USART\_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

*Note:*

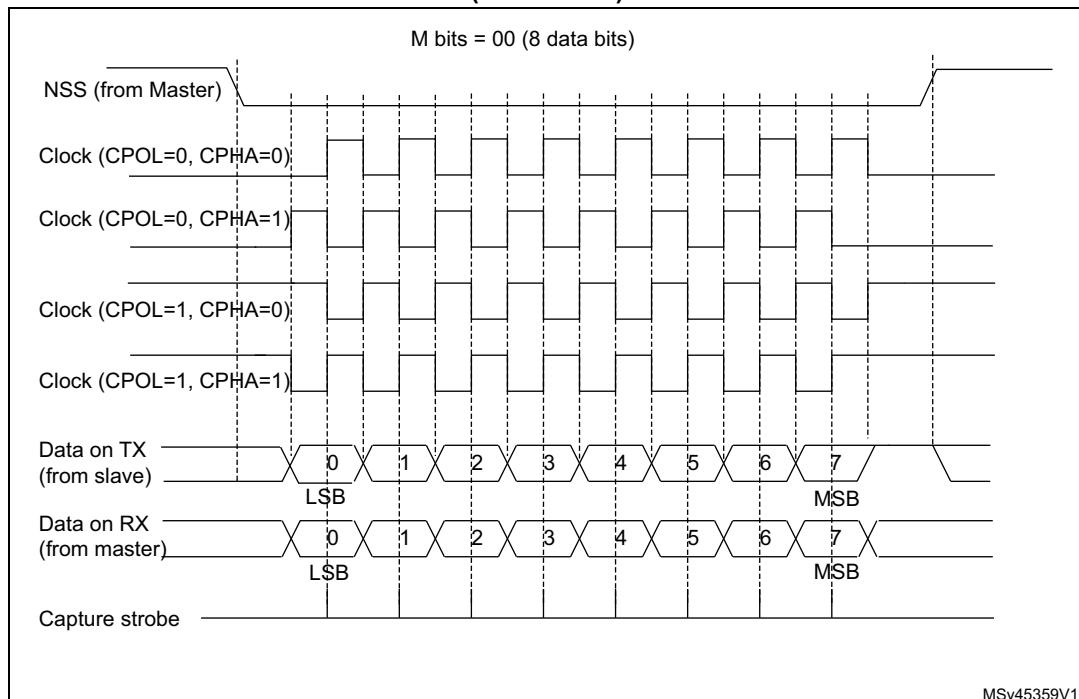
*When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (uart\_ker\_ck\_pres) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART\_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see [Figure 335](#)).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART\_TDR.

The slave supports the hardware and software NSS management.

**Figure 335. USART data clock timing diagram in synchronous slave mode  
(M bits = 00)**



### Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS\_NSS bit in the USART\_CR2 register:

- Software NSS management (DIS\_NSS = 1)  
The SPI slave is always selected and NSS input pin is ignored.  
The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS\_NSS = 0)  
The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

*Note:* The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.

*In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.*

### SPI Slave underrun error

When an underrun error occurs, the UDR flag is set in the USART\_ISR register, and the SPI slave goes on sending the last data until the underrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART\_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART\_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

*Note:* *An underrun error may occur if the moment the data is written to the USART\_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART\_TDR should be written 3 usart\_ker\_ck cycles before the first CK edge.*

### 33.5.15 USART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the USART\_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART\_CR2 register,
- SCEN and IREN bits in the USART\_CR3 register.

The USART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in USART\_CR3.

As soon as HDSEL is written to '1':

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

### 33.5.16 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART\_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART\_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = '00' or STOP = '11'
- from the end of the second stop bit if STOP = '10'.
- from the beginning of the stop bit if STOP = '01'.

When the timeout duration has elapsed, the RTOF flag in the USART\_ISR register is set. A timeout is generated if RTOIE bit in USART\_CR1 register is set.

### 33.5.17 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to [Section 33.4: USART implementation on page 1038](#).

Smartcard mode is selected by setting the SCEN bit in the USART\_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART\_CR2 register,
- HDSEL and IREN bits in the USART\_CR3 register.

The CLKEN bit can also be set to provide a clock to the Smartcard.

The Smartcard interface is designed to support asynchronous Smartcard protocol as defined in the ISO 7816-3 standard. Both  $T = 0$  (character mode) and  $T = 1$  (block mode) are supported.

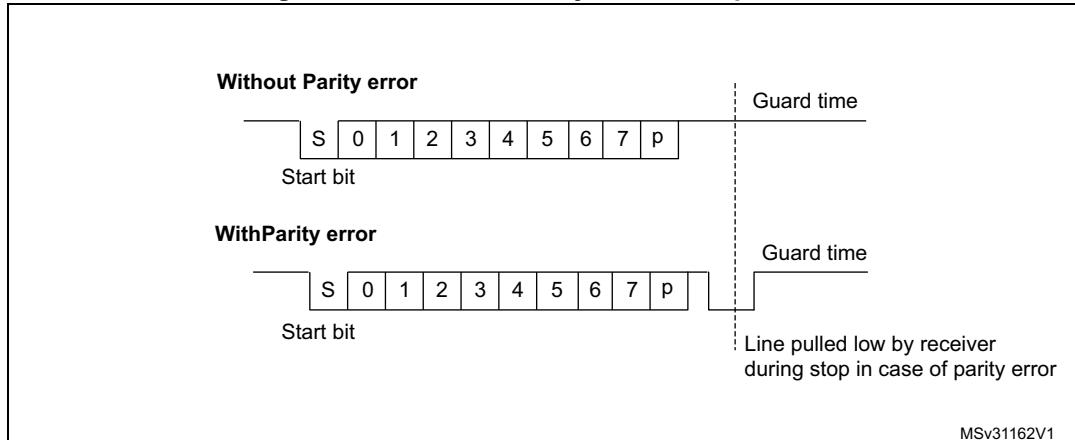
The USART should be configured as:

- 8 bits plus parity:  $M = 1$  and  $PCE = 1$  in the USART\_CR1 register
- 1.5 stop bits when transmitting and receiving data:  $STOP = '11'$  in the USART\_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In  $T = 0$  (character) mode, the parity error is indicated at the end of each character during the guard time period.

[Figure 336](#) shows examples of what can be seen on the data line with and without parity error.

**Figure 336. ISO 7816-3 asynchronous protocol**



MSv31162V1

When connected to a Smartcard, the TX output of the USART drives a bidirectional line that is also driven by the Smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the Smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART\_RQR register.

- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the Smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The deassertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

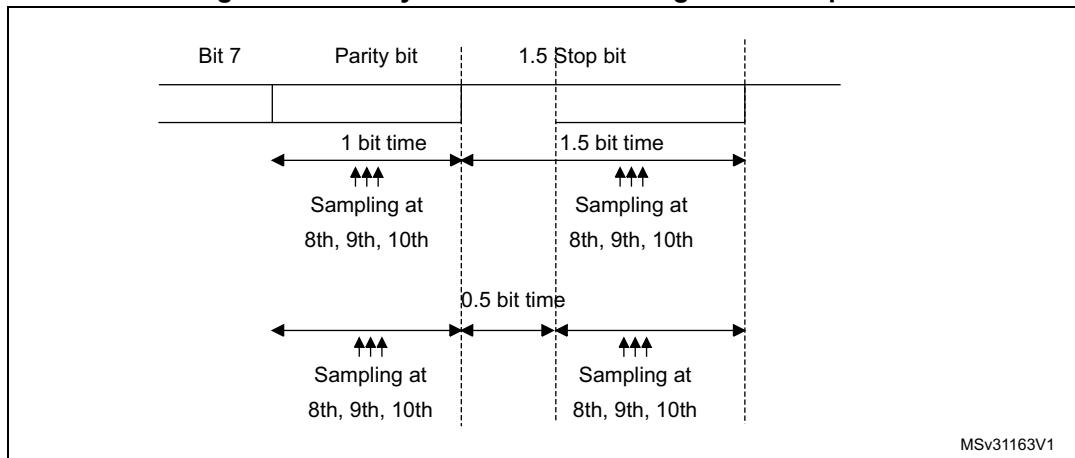
*Note:*

*Break characters are not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.*

*No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.*

*Figure 337* shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 337. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the Smartcard through the CK output. In Smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART\_GTPR register. CK frequency can be programmed from `usart_ker_ck_pres/2` to `usart_ker_ck_pres/62`, where `usart_ker_ck_pres` is the peripheral input clock divided by a programmed prescaler.

### Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART\_CR3 register.

When requesting a read from the Smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

**Note:** *The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the Smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.*

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time -11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the Smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

**Note:** *As in the Smartcard protocol definition, the BWT/CWT values should be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT -11, respectively, taking into account the length of the last character itself.*

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the Smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART\_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN = LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN = LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBI bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

*Note:* *The error checking code (LRC/CRC) must be computed/verified by software.*

### Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV = 1.

*Note:* *When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.*

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, assuming that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH results in a USART received character of 03 and an odd parity.

Therefore, two methods are available for TS pattern recognition:

#### Method 1

The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART.

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

#### Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

- (H) LHHL LLL LLH = 0x103: inverse convention to be chosen
- (H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

### 33.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to [Section 33.4: USART implementation on page 1038](#).

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 338](#)).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbaud for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not

encoded. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.

- A '0' is transmitted as a high pulse and a '1' is transmitted as a '0'. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see [Figure 339](#)).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41  $\mu$ s. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART\_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART\_CR2 register must be configured to '1 stop bit'.

### IrDA low-power mode

- Transmitter

In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.

- Receiver

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the USART\_GTPR).

**Note:** *A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.*

*The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).*

Figure 338. IrDA SIR ENDEC block diagram

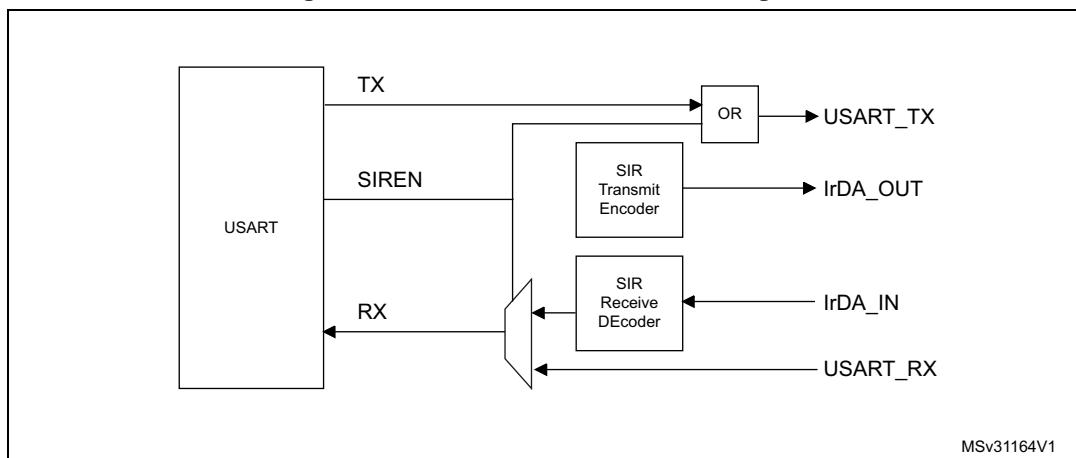
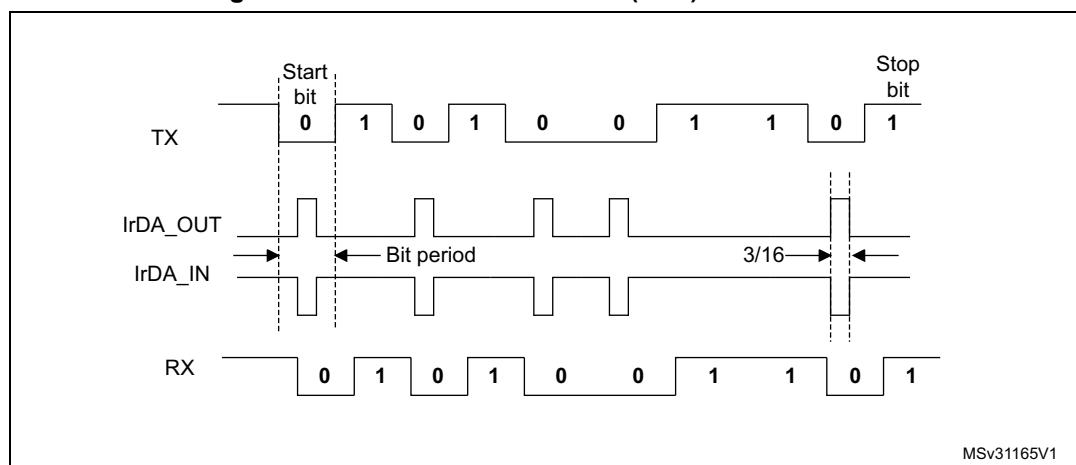


Figure 339. IrDA data modulation (3/16) - Normal mode



### 33.5.19 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

**Note:** Refer to [Section 33.4: USART implementation on page 1038](#) to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in [Section 33.5.6](#). To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART\_ISR register.

#### Transmission using DMA

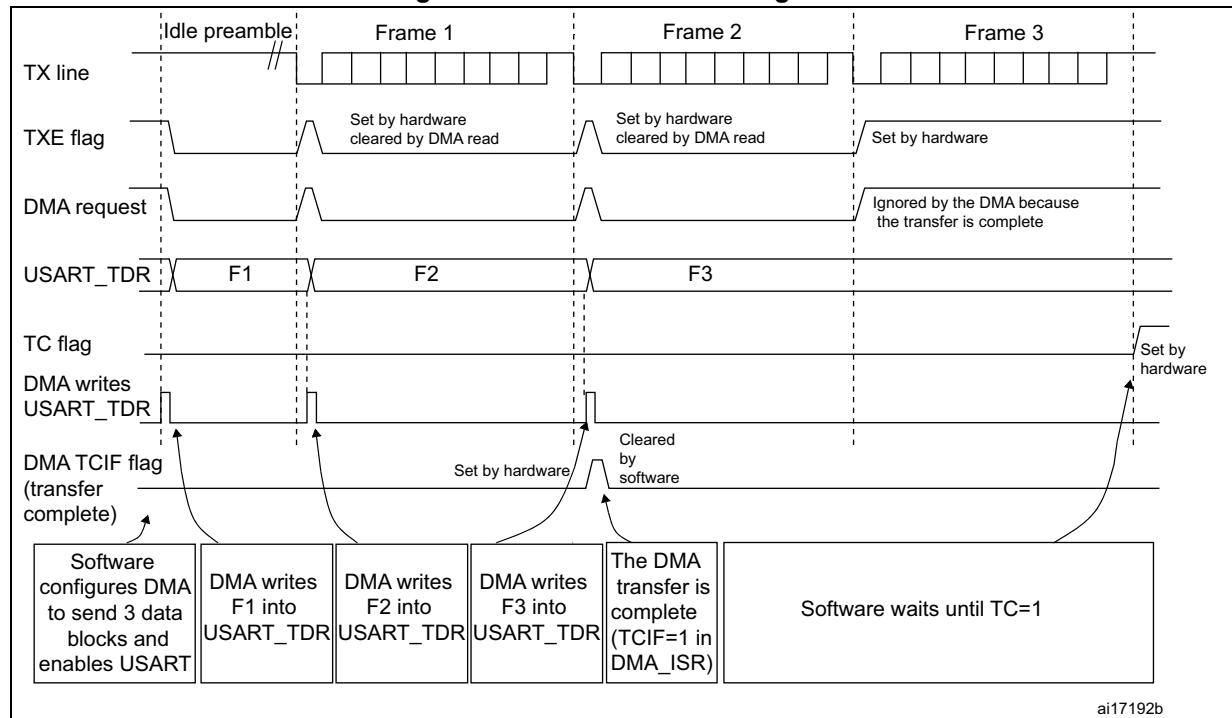
DMA mode can be enabled for transmission by setting DMAT bit in the USART\_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the USART\_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART\_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART\_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART\_ISR register by setting the TCCF bit in the USART\_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA\_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 340. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

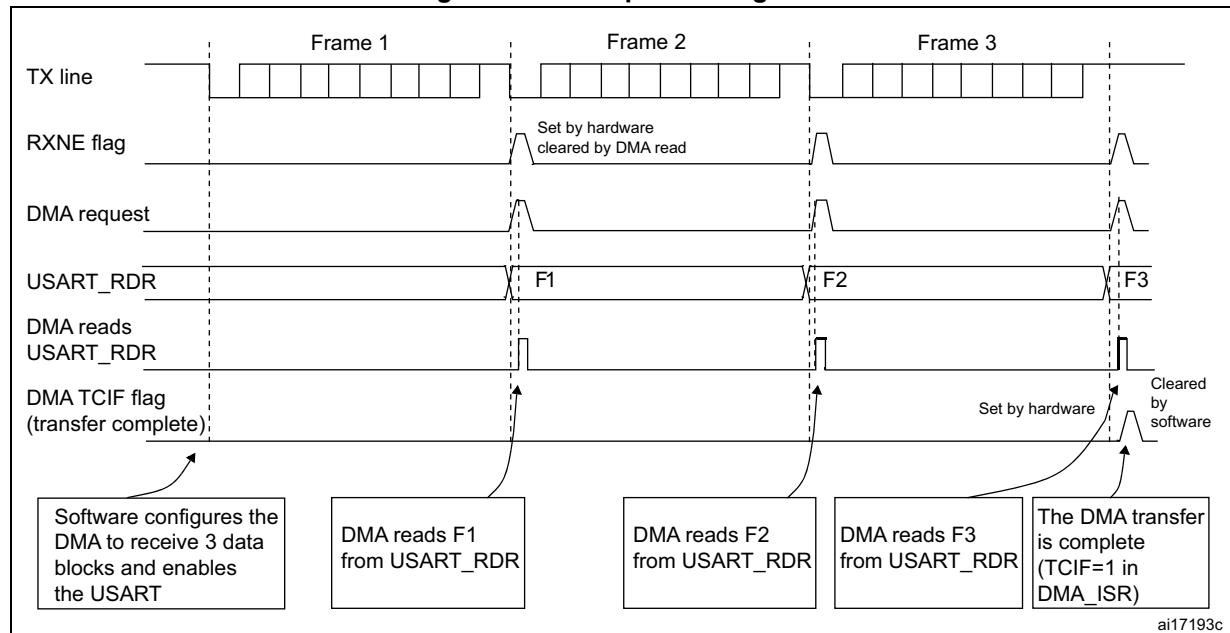
### Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART\_CR3 register. Data are loaded from the USART\_RDR register to an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART\_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART\_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 341. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

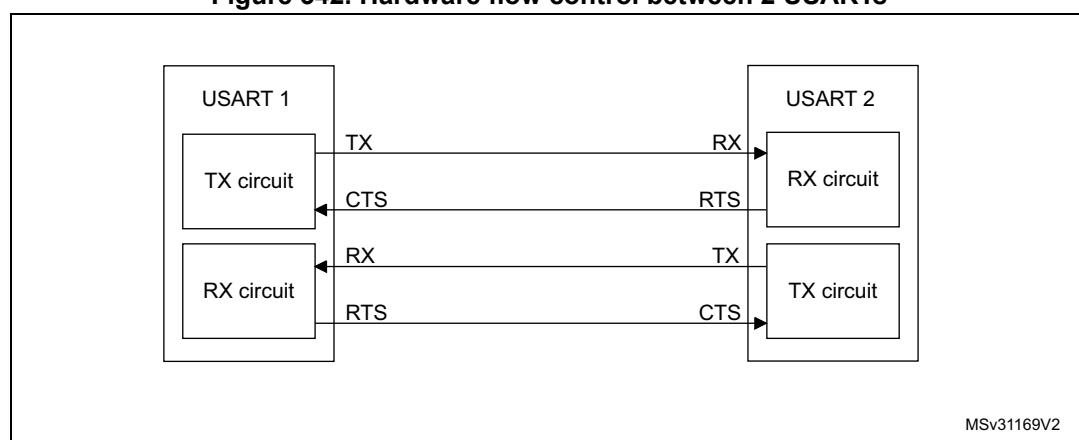
#### Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART\_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

#### 33.5.20 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The [Figure 342](#) shows how to connect 2 devices in this mode:

Figure 342. Hardware flow control between 2 USARTs

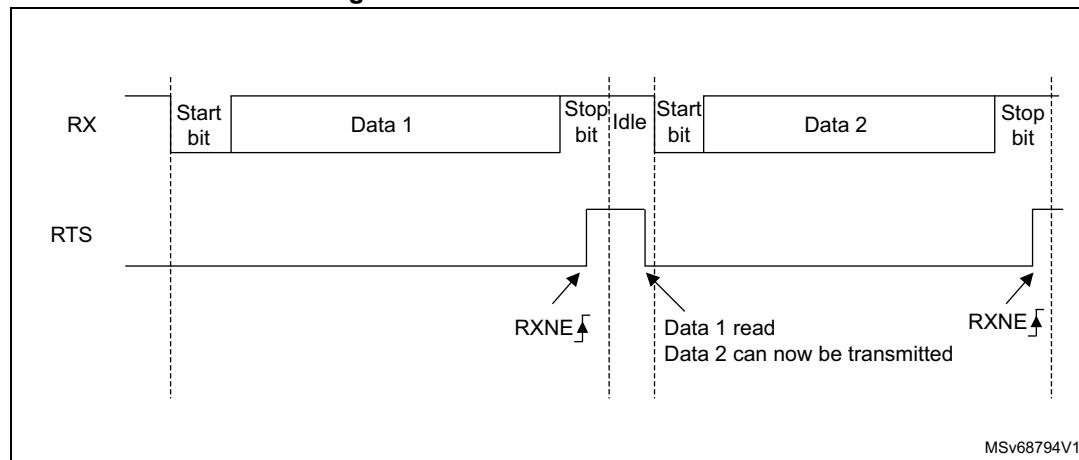


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to '1' in the USART\_CR3 register.

### RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. [Figure 343](#) shows an example of communication with RTS flow control enabled.

**Figure 343. RS232 RTS flow control**



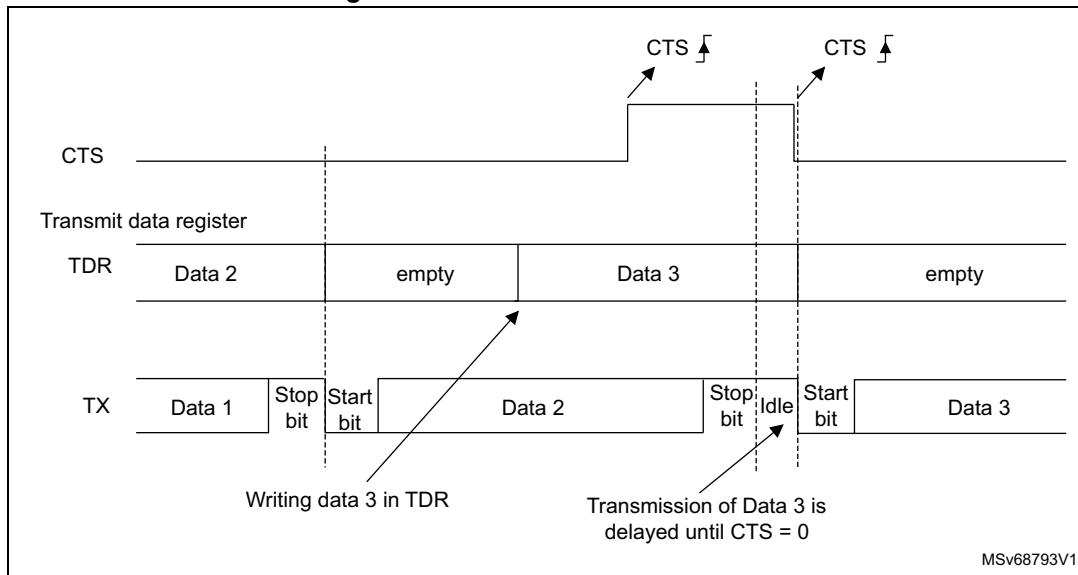
**Note:** When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

### RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART\_CR3 register is set. [Figure 344](#) shows an example of communication with CTS flow control enabled.

Figure 344. RS232 CTS flow control



**Note:** For correct behavior, CTS must be deasserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

### RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART\_CR3 control register. This enables the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the USART\_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART\_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART\_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

### 33.5.21 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the `usart_pclk` clock is disabled.

The USART is able to wake up the MCU from low-power mode when the `UESM` bit is set.

When the `usart_pclk` is gated, the USART provides a wakeup interrupt (`usart_wkup`) if a specific action requiring the activation of the `usart_pclk` clock is needed:

- If FIFO mode is disabled
  - `usart_pclk` clock has to be activated to empty the USART data register.
  - In this case, the `usart_wkup` interrupt source is `RXNE` set to '1'. The `RXNEIE` bit must be set before entering low-power mode.
- If FIFO mode is enabled
  - `usart_pclk` clock has to be activated to:
    - to fill the TXFIFO
    - or to empty the RXFIFO
  - In this case, the `usart_wkup` interrupt source can be:
    - RXFIFO not empty. In this case, the `RXFNEIE` bit must be set before entering low-power mode.
    - RXFIFO full. In this case, the `RXFFIE` bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the `RXFF` flag is not set.
    - TXFIFO empty. In this case, the `TXFEIE` bit must be set before entering low-power mode.

This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

To avoid overrun/underrun errors and transmit/receive data in low-power mode, the `usart_wkup` interrupt source can be one of the following events:

- TXFIFO threshold reached. In this case, the `TXFTIE` bit must be set before entering low-power mode.
- RXFIFO threshold reached. In this case, the `RXFTIE` bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time required to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific `usart_wkup` interrupt can be selected through the `WUS` bitfields.

When the wakeup event is detected, the `WUF` flag is set by hardware and a `usart_wkup` interrupt is generated if the `WUFIE` bit is set.

- Note:** *Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.*
- The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in low-power or active mode.*
- When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.*
- When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.*
- When the FIFO is enabled, waking up from low-power mode on address match is only possible when Mute mode is enabled.*

### Using Mute mode with low-power mode

If the USART is put into Mute mode before entering low-power mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wakeup from Mute mode on address match is used, then the low-power mode wakeup source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in Mute mode upon address match and wake up from low-power mode.

- Note:** *When FIFO management is enabled, Mute mode can be used with wakeup from low-power mode without any constraints (i.e. the two points mentioned above about Mute and low-power mode are valid only when FIFO management is disabled).*

### Wakeup from low-power mode when USART kernel clock (usart\_ker\_ck) is OFF in low-power mode

If during low-power mode, the usart\_ker\_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart\_ker\_ck clock to be switched ON thanks to the usart\_ker\_ck\_req signal. usart\_ker\_ck is then used for the frame reception.

If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wakeup event is not verified, usart\_ker\_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wakeup event programmed to “address match detection” and FIFO management disabled.

Figure 345 shows the USART behavior when the wakeup event is verified.

**Figure 345. Wakeup event verified (wakeup event = address match, FIFO disabled)**

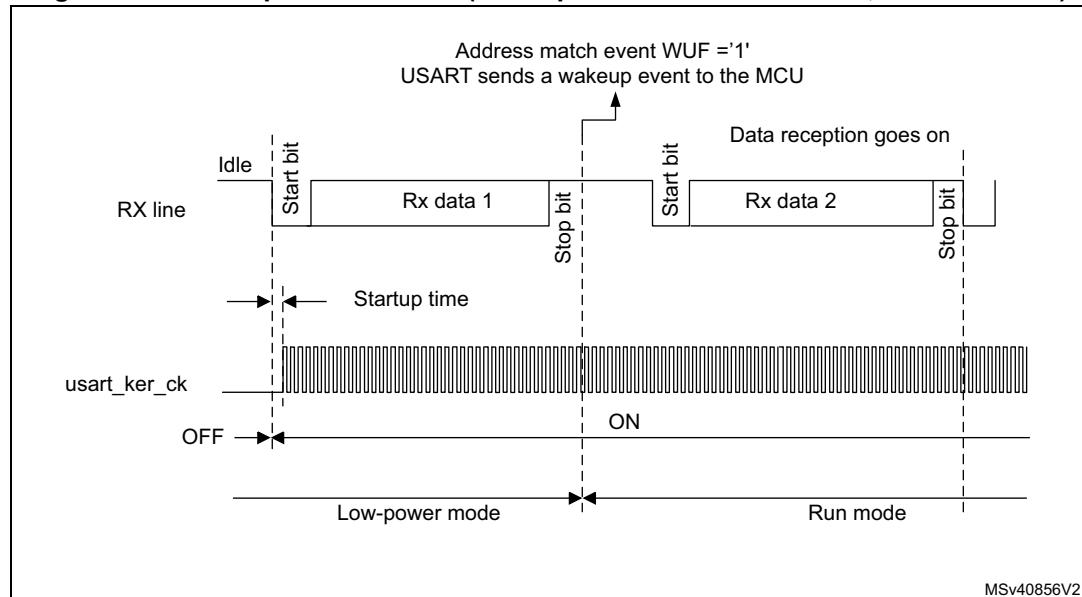
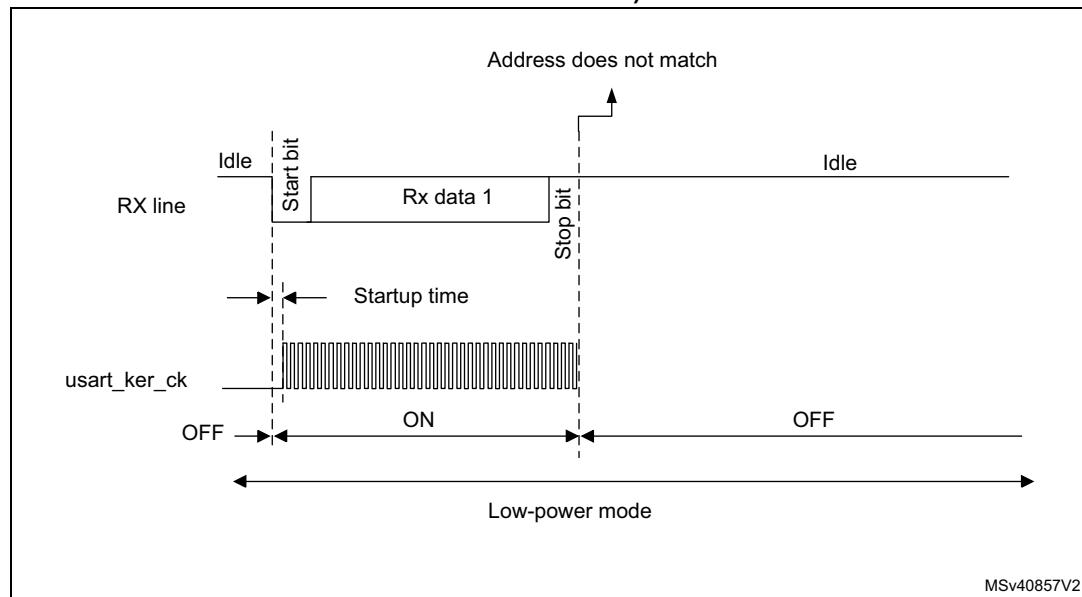


Figure 346 shows the USART behavior when the wakeup event is not verified.

**Figure 346. Wakeup event not verified (wakeup event = address match, FIFO disabled)**



Note:

The figures above are valid when address match or any received frame is used as wakeup event. If the wakeup event is the start bit detection, the USART sends the wakeup event to the MCU at the end of the start bit.

### Determining the maximum USART baud rate that enables to correctly wake up the device from low-power mode

The maximum baud rate that enables to correctly wake up the device from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the USART receiver tolerance (see [Section 33.5.8: Tolerance of the USART receiver to clock deviation](#)).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to [Table 206: Tolerance of the USART receiver when BRR \[3:0\] = 0000](#), the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$$D_{WUmax} = t_{WUUSART} / (11 \times T_{bit\ Min})$$

$$T_{bit\ Min} = t_{WUUSART} / (11 \times D_{WUmax})$$

where  $t_{WUUSART}$  is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the usart\_ker\_ck inaccuracy.

For example, if HSI is used as usart\_ker\_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART} = 3\ \mu s$  (values provided only as examples; for correct values, refer to the device datasheet).

$$D_{WUmax} = 3.41\% - 1\% = 2.41\%$$

$$T_{bit\ min} = 3\ \mu s / (11 \times 2.41\%) = 11.32\ \mu s$$

As a result, the maximum baud rate that enables to wakeup correctly from low-power mode is:  $1/11.32\ \mu s = 88.36\ Kbaud$ .

## 33.6 USART in low-power modes

**Table 209. Effect of low-power modes on the USART**

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop <sup>(1)</sup>	The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to [Section 33.4: USART implementation](#) to know if the wakeup from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

## 33.7 USART interrupts

Refer to [Table 210](#) for a detailed description of all USART interrupt requests.

**Table 210. USART interrupt requests**

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop <sup>(1)</sup> modes	Exit from Standby mode
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Transmission Complete Before Guard Time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	
USART or UART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	No
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF <sup>(2)</sup>	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RXNEIE/RXFNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication	ORE <sup>(3)</sup>		Write 1 in ORECF		No	
	Framing Error in multibuffer communication	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	Wakeup from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

- The USART can wake up the device from Stop mode only if the peripheral instance supports the Wakeup from Stop mode feature. Refer to [Section 33.4: USART implementation](#) for the list of supported Stop modes.

2. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART\_RDR. In Stop mode, USART\_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

## 33.8 USART registers

Refer to [Section 1.2 on page 61](#) for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

### 33.8.1 USART control register 1 (USART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

#### FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFIE	TXFEIE	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]						DEDT[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFI E	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when RXFF = 1 in the USART\_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when TXFE = 1 in the USART\_ISR register

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.*

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = '00': 1 start bit, 8 Data bits, n Stop bit

M[1:0] = '01': 1 start bit, 9 Data bits, n Stop bit

M[1:0] = '10': 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE = 0).

*Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.*

Bit 27 **EOBIE**: End-of-block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART\_ISR register

*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART\_ISR register.

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. [Section 33.4: USART implementation on page 1038](#).*

Bits 25:21 **DEAT[4:0]**: Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bits 20:16 **DEDT[4:0]**: Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART\_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

*Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.*

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART\_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART\_ISR register

Bit 7 **TXFNIE**: TXFIFO not-full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXFNF = 1 in the USART\_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART\_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART\_ISR register

**Bit 4 IDLEIE:** IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART\_ISR register

**Bit 3 TE:** Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

*Note: During transmission, a low pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1'. To ensure the required duration, the software can poll the TEACK bit in the USART\_ISR register.*

*In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.*

**Bit 2 RE:** Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

**Bit 1 UESM:** USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

*Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.*

*If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

**Bit 0 UE:** USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART\_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART\_ISR to be set before resetting the UE bit.*

*The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.*

*In Smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.*

### 33.8.2 USART control register 1 [alternate] (USART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

#### FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

#### Bit 29 FIFOEN: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.*

#### Bit 28 M1: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = '00': 1 start bit, 8 Data bits, n Stop bit

M[1:0] = '01': 1 start bit, 9 Data bits, n Stop bit

M[1:0] = '10': 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE = 0).

*Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.*

#### Bit 27 EOBIE: End of Block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART\_ISR register

*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

#### Bit 26 RTOIE: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART\_ISR register.

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 33.4: USART implementation on page 1038.*

Bits 25:21 **DEAT[4:0]**: Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bits 20:16 **DEDT[4:0]**: Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART\_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

*Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.*

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART\_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART\_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXE = 1 in the USART\_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART\_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXNE = 1 in the USART\_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART\_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

*Note: During transmission, a low pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1'. To ensure the required duration, the software can poll the TEACK bit in the USART\_ISR register.*

*In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.*

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

*Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.*

*If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART\_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART\_ISR to be set before resetting the UE bit.*

*The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.*

*In Smartcard mode, (SCEN = 1), the CK pin is always available when CLKEN = 1, regardless of the UE bit value.*

### 33.8.3 USART control register 2 (USART\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								RTOEN	ABRMOD[1:0]	ABREN	MSBF1 RST	DATAINV	TXINV	RXINV	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]	CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN	
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw	

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in Mute mode or a character code to be recognized in low-power or Run mode:

- In Mute mode: they are used in multiprocessor communication to wakeup from Mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wakeup from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with Mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART\_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 and Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

*Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)*

*If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 18 DATAINV: Binary data inversion**

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H).

The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 17 TXINV: TX pin active level inversion**

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels ( $V_{DD} = 1/\text{idle}$ , Gnd = 0/mark)

1: TX pin signal values are inverted ( $V_{DD} = 0/\text{mark}$ , Gnd = 1/idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 16 RXINV: RX pin active level inversion**

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels ( $V_{DD} = 1/\text{idle}$ , Gnd = 0/mark)

1: RX pin signal values are inverted ( $V_{DD} = 0/\text{mark}$ , Gnd = 1/idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 15 SWAP: Swap TX/RX pins**

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another USART.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 14 LINEN: LIN mode enable**

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART\_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value.*

*Refer to [Section 33.4: USART implementation on page 1038](#).*

**Bits 13:12 STOP[1:0]: Stop bits**

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

This bit can only be written when the USART is disabled (UE = 0).

*Note: If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

*In Smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:*

*UE = 0*

*SCEN = 1*

*GTPR configuration*

*CLKEN= 1*

*UE = 1*

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

*Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see [Figure 326](#) and [Figure 327](#))

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

*Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

**Caution:** The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART\_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

*Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

## Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART\_ISR register

*Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE = 0).

*Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

Bit 4 **ADDM7**: 7-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE = 0)

*Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.*

Bit 3 **DIS\_NSS**: NSS pin enable

When the DIS\_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.

1: SPI slave is always selected and NSS input pin is ignored.

*Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous Slave mode enable

When the SLVEN bit is set, the synchronous slave mode is enabled.

0: Slave mode disabled.

1: Slave mode enabled.

*Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

*Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.*

### 33.8.4 USART control register 3 (USART\_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS[1:0]		SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]**: TXFIFO threshold configuration

- 000:TXFIFO reaches 1/8 of its depth
- 001:TXFIFO reaches 1/4 of its depth
- 010:TXFIFO reaches 1/2 of its depth
- 011:TXFIFO reaches 3/4 of its depth
- 100:TXFIFO reaches 7/8 of its depth
- 101:TXFIFO becomes empty
- Remaining combinations: Reserved

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration

- 000:Receive FIFO reaches 1/8 of its depth
- 001:Receive FIFO reaches 1/4 of its depth
- 010:Receive FIFO reaches 1/2 of its depth
- 011:Receive FIFO reaches 3/4 of its depth
- 100:Receive FIFO reaches 7/8 of its depth
- 101:Receive FIFO becomes full
- Remaining combinations: Reserved

Bit 24 **TCBGTIE**: Transmission complete before guard time, interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated whenever TCBGT=1 in the USART\_ISR register

*Note: If the USART does not support the Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from low-power mode interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated whenever WUF = 1 in the USART\_ISR register

*Note: WUFIE must be set before entering in low-power mode.*

*If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bits 21:20 **WUS[1:0]**: Wakeup from low-power mode interrupt flag selection  
 This bitfield specifies the event which activates the WUF (Wakeup from low-power mode flag).

- 00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
- 01: Reserved.
- 10: WUF active on start bit detection
- 11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE = 0).

*If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in Smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number of erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmit mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

*Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

- 0: DE signal is active high.
- 1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.  
 0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 13 **DDRE**: DMA Disable on reception error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred (used for Smartcard mode).

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE/RXFNE in case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE=0).

*Note: The reception errors are: parity error, framing error or noise error.*

Bit 12 **OVRDIS**: Overrun disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART\_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data is written directly in USART\_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

*Note: This control bit enables checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART\_ISR register

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0).

If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling Smartcard mode.

0: Smartcard Mode disabled

1: Smartcard Mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

*Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

*Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART\_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART\_ISR register.

### 33.8.5 USART baud rate register (USART\_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]**: USART baud rate

**BRR[15:4]**

BRR[15:4] = USARTDIV[15:4]

**BRR[3:0]**

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

### 33.8.6 USART guard time and prescaler register (USART\_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
GT[7:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bits 7:0 **PSC[7:0]**: Prescaler value

**In IrDA low-power and normal IrDA mode:**

PSC[7:0] = IrDA Normal and Low-Power baud rate

PSC[7:0] is used to program the prescaler for dividing the USART source clock to achieve the low-power frequency: the source clock is divided by the value given in the register (8 significant bits):

**In Smartcard mode:**

PSC[4:0] = Prescaler value

PSC[4:0] is used to program the prescaler for dividing the USART source clock to provide the Smartcard clock. The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: Divides the source clock by 1 (IrDA mode) / by 2 (Smartcard mode)

00010: Divides the source clock by 2 (IrDA mode) / by 4 (Smartcard mode)

00011: Divides the source clock by 3 (IrDA mode) / by 6 (Smartcard mode)

...

11111: Divides the source clock by 31 (IrDA mode) / by 62 (Smartcard mode)

0010 0000: Divides the source clock by 32 (IrDA mode)

...

1111 1111: Divides the source clock by 255 (IrDA mode)

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: Bits [7:5] must be kept cleared if Smartcard mode is used.*

*This bitfield is reserved and forced by hardware to '0' when the Smartcard and IrDA modes are not supported. Refer to [Section 33.4: USART implementation on page 1038](#).*

### 33.8.7 USART receiver timeout register (USART\_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]								RTO[15:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block length

This bitfield gives the Block length in Smartcard T = 1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

BLEN = 0: 0 information characters + LEC

BLEN = 1: 0 information characters + CRC

BLEN = 255: 254 information characters + CRC (total 256 characters)

In Smartcard mode, the Block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the Block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

*Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.*

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bits during which there is no activity on the RX line.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

*Note: This value must only be programmed once per received character.*

*Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.*

*This register is reserved and forced by hardware to “0x00000000” when the Receiver timeout feature is not supported. Refer to [Section 33.4: USART implementation on page 1038](#).*

### 33.8.8 USART request register (USART\_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ										
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

**Bit 4 TXFRQ:** Transmit data flush request

When FIFO mode is disabled, writing '1' to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in Smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART\_ISR register. If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART\_ISR register). Flushing the Transmit FIFO is supported in both UART and Smartcard modes.

*Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.*

**Bit 3 RXFRQ:** Receive data flush request

Writing 1 to this bit empties the entire receive FIFO i.e. clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

**Bit 2 MMRQ:** Mute mode request

Writing 1 to this bit puts the USART in Mute mode and resets the RWU flag.

**Bit 1 SBKRQ:** Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

*Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.*

**Bit 0 ABRRQ:** Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART\_ISR and requests an automatic baud rate measurement on the next received data frame.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

### 33.8.9 USART interrupt and status register (USART\_ISR)

Address offset: 0x1C

Reset value: 0x0X80 00C0

X = 2 if FIFO/Smartcard mode is enabled

X = 0 if FIFO is enabled and Smartcard mode is disabled

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

#### FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG of USART\_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the USART\_CR3 register.

- 0: TXFIFO does not reach the programmed threshold.
- 1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in RXFTCFG in USART\_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART\_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the USART\_CR3 register.

- 0: Receive FIFO does not reach the programmed threshold.
- 1: Receive FIFO reached the programmed threshold.

*Note: When the RXFTCFG threshold is configured to '101', RXFT flag is set if 16 data are available i.e. 15 data in the RXFIFO and 1 data in the USART\_RDR. Consequently, the 17th received data does not cause an overrun error. The overrun error occurs after receiving the 18th data.*

Bit 25 **TCBGT**: Transmission complete before guard time flag

This bit is set when the last data written in the USART\_TDR has been transmitted correctly out of the shift register.

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART\_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART\_ICR register or by a write to the USART\_TDR register.

- 0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)
- 1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

*Note: If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is '1'. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 24 **RXFF**: RXFIFO full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART\_RDR register).

An interrupt is generated if the RXFFIE bit = 1 in the USART\_CR1 register.

- 0: RXFIFO not full.
- 1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO empty

This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART\_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART\_CR1 register.

- 0: TXFIFO not empty.
- 1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

*Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART\_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART\_ICR register.

An interrupt is generated if WUFIE = 1 in the USART\_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*

*If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

Bit 19 **RWU**: Receiver wakeup from Mute mode

This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART\_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART\_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

*Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART\_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: Break character transmitted

1: Break character requested by setting SBKRQ bit in USART\_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART\_ICR register.

An interrupt is generated if CMIE = 1 in the USART\_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF**: Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation was completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case). It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART\_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART\_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART\_TDR. This flag is reset by setting UDRCF bit in the USART\_ICR register.

0: No underrun error

1: underrun error

*Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBIIE = 1 in the USART\_CR1 register.

It is cleared by software, writing 1 to the EOBCF in the USART\_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

*Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART\_ICR register.

An interrupt is generated if RTOIE = 1 in the USART\_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

*Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.*

*The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.*

*If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.*

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART\_ICR register.

An interrupt is generated if CTSIE = 1 in the USART\_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART\_ICR.

An interrupt is generated if LBDIE = 1 in the USART\_CR2 register.

0: LIN Break not detected

1: LIN break detected

*Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 7 **TXFNF**: TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART\_TDR. Every write operation to the USART\_TDR places the data in the TXFIFO.

This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART\_TDR.

An interrupt is generated if the TXFNIE bit =1 in the USART\_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

*Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).*

*This bit is used during single buffer transmission.*

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART\_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXFE is set.

An interrupt is generated if TCIE = 1 in the USART\_CR1 register.

TC bit is cleared by software, by writing 1 to the TCCF in the USART\_ICR register or by a write to the USART\_TDR register.

0: Transmission is not complete

1: Transmission is complete

*Note: If TE bit is reset and no transmission is on going, the TC bit is immediately set.*

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART\_RDR register. Every read operation from the USART\_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART\_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART\_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART\_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART\_ICR register.

0: No Idle line is detected

1: Idle line is detected

*Note: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).*

*If Mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the Mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.*

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART\_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART\_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART\_CR1 register, or EIE = 1 in the USART\_CR3 register.

0: No overrun error

1: Overrun error is detected

*Note: When this bit is set, the USART\_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

*This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART\_CR3 register.*

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART\_ICR register.

0: No noise is detected

1: Noise is detected

*Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.*

*When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to [Section 33.5.8: Tolerance of the USART receiver to clock deviation on page 1055](#)).*

*This error is associated with the character in the USART\_RDR.*

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART\_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART\_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

*Note: This error is associated with the character in the USART\_RDR.*

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECE bit in the USART\_ICR register.

An interrupt is generated if PEIE = 1 in the USART\_CR1 register.

0: No parity error

1: Parity error

*Note: This error is associated with the character in the USART\_RDR.*

### 33.8.10 USART interrupt and status register [alternate] (USART\_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

#### FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT**: Transmission complete before guard time flag

This bit is set when the last data written in the USART\_TDR has been transmitted correctly out of the shift register.

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART\_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART\_ICR register or by a write to the USART\_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

*Note: If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is '1'. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

*Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART\_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART\_ICR register. An interrupt is generated if WUFIE = 1 in the USART\_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*

*If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 19 **RWU**: Receiver wakeup from Mute mode

This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART\_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART\_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

*Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART\_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: Break character transmitted

1: Break character requested by setting SBKRQ bit in USART\_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART\_ICR register.

An interrupt is generated if CMIE = 1 in the USART\_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF**: Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART\_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART\_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART\_TDR. This flag is reset by setting UDRCF bit in the USART\_ICR register.

0: No underrun error

1: underrun error

*Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBIIE = 1 in the USART\_CR1 register.

It is cleared by software, writing 1 to the EOBCF in the USART\_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

*Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART\_ICR register.

An interrupt is generated if RTOIE = 1 in the USART\_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

*Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.*

*The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.*

*If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.*

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART\_ICR register.

An interrupt is generated if CTSIE = 1 in the USART\_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART\_ICR.

An interrupt is generated if LBDIE = 1 in the USART\_CR2 register.

0: LIN Break not detected

1: LIN break detected

*Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value.  
Refer to [Section 33.4: USART implementation on page 1038](#).*

Bit 7 **TXE**: Transmit data register empty

TXE is set by hardware when the content of the USART\_TDR register has been transferred into the shift register. It is cleared by writing to the USART\_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART\_RQR register, in order to discard the data (only in Smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXIE bit = 1 in the USART\_CR1 register.

0: Data register full

1: Data register not full

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART\_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART\_CR1 register.

TC bit is cleared by software, by writing 1 to the TCCF in the USART\_ICR register or by a write to the USART\_TDR register.

0: Transmission is not complete

1: Transmission is complete

*Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.*

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART\_RDR shift register has been transferred to the USART\_RDR register. It is cleared by reading from the USART\_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART\_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART\_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART\_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART\_ICR register.

0: No Idle line is detected

1: Idle line is detected

*Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).*

*If Mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the Mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.*

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART\_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART\_ICR register.

An interrupt is generated if RXNEIE = 1 or EIE = 1 in the LPUART\_CR1 register, or EIE = 1 in the USART\_CR3 register.

0: No overrun error

1: Overrun error is detected

*Note: When this bit is set, the USART\_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

*This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART\_CR3 register.*

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART\_ICR register.

- 0: No noise is detected  
1: Noise is detected

*Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.*

*When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to [Section 33.5.8: Tolerance of the USART receiver to clock deviation on page 1055](#)).*

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART\_ICR register. When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

- An interrupt is generated if EIE = 1 in the USART\_CR3 register.  
0: No Framing error is detected  
1: Framing error or break character is detected

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART\_ICR register.

- An interrupt is generated if PEIE = 1 in the USART\_CR1 register.  
0: No parity error  
1: Parity error

### 33.8.11 USART interrupt flag clear register (USART\_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGT CF	TCCF	TXFEC F	IDLEC F	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART\_ISR register.

*Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#).*

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART\_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART\_ISR register.

*Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#)*

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART\_ISR register.

*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#)*

Bit 11 **RTOCF**: Receiver timeout clear flag

Writing 1 to this bit clears the RTOF flag in the USART\_ISR register.

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#)*

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the USART\_ISR register.

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#)*

Bit 8 **LBDCF**: LIN break detection clear flag

Writing 1 to this bit clears the LBDF flag in the USART\_ISR register.

*Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 1038](#)*

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag

Writing 1 to this bit clears the TCBGT flag in the USART\_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the USART\_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag

Writing 1 to this bit clears the TXFE flag in the USART\_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag

Writing 1 to this bit clears the IDLE flag in the USART\_ISR register.

Bit 3 **ORECF**: Overrun error clear flag

Writing 1 to this bit clears the ORE flag in the USART\_ISR register.

Bit 2 **NECF**: Noise detected clear flag

Writing 1 to this bit clears the NE flag in the USART\_ISR register.

Bit 1 **FECF**: Framing error clear flag

Writing 1 to this bit clears the FE flag in the USART\_ISR register.

Bit 0 **PECF**: Parity error clear flag

Writing 1 to this bit clears the PE flag in the USART\_ISR register.

### 33.8.12 USART receive data register (USART\_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RDR[8:0]														
								r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see [Figure 320](#)).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

### 33.8.13 USART transmit data register (USART\_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDR[8:0]														
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART\_TDR register provides the parallel interface between the internal bus and the output shift register (see [Figure 320](#)).

When transmitting with the parity enabled (PCE bit set to 1 in the USART\_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

*Note: This register must be written only when TXE/TXFNF = 1.*

### 33.8.14 USART prescaler register (USART\_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PRESCALER[3:0]														
															rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

- 0000: input clock not divided
- 0001: input clock divided by 2
- 0010: input clock divided by 4
- 0011: input clock divided by 6
- 0100: input clock divided by 8
- 0101: input clock divided by 10
- 0110: input clock divided by 12
- 0111: input clock divided by 16
- 1000: input clock divided by 32
- 1001: input clock divided by 64
- 1010: input clock divided by 128
- 1011: input clock divided by 256

Remaining combinations: Reserved

*Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is 1011 i.e. input clock divided by 256.*

### 33.8.15 USART register map

The table below gives the USART register map and reset values.

Table 211. USART register map and reset values

Offset	Register name	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	
0x00	USART_CR1 FIFO enabled	0	RXFFIE	0	TXFEIE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	0	Res.	Res.	Res.	M1	0	M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x00	USART_CR1 FIFO disabled	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
0x04	USART_CR2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x08	USART_CR3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C	USART_BRR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
0x10	USART_GTPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
0x14	USART_RTOR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x1C	USART_ISR FIFO mode enabled	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x1C	USART_ISR FIFO mode disabled	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x20	USART_ICR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x24	USART_RDR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 211. USART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	USART_TDR	Res.																															
	Reset value																																
0x2C	USART_PRESC	Res.																															
	Reset value																																

Refer to [Section 2.2: Memory organization](#) for the register boundary addresses.

## 34 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitted (LPUART).

### 34.1 LPUART introduction

The LPUART is an UART which enables bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to enable UART communications up to 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the device is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports Half-duplex Single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

## 34.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 baud to 9600 baud using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data
  - Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wakeup from Mute mode by idle line detection or address mark detection

### 34.3 LPUART implementation

Below the description of LPUART implementation in comparison with USART.

**Table 212. USART / LPUART features**

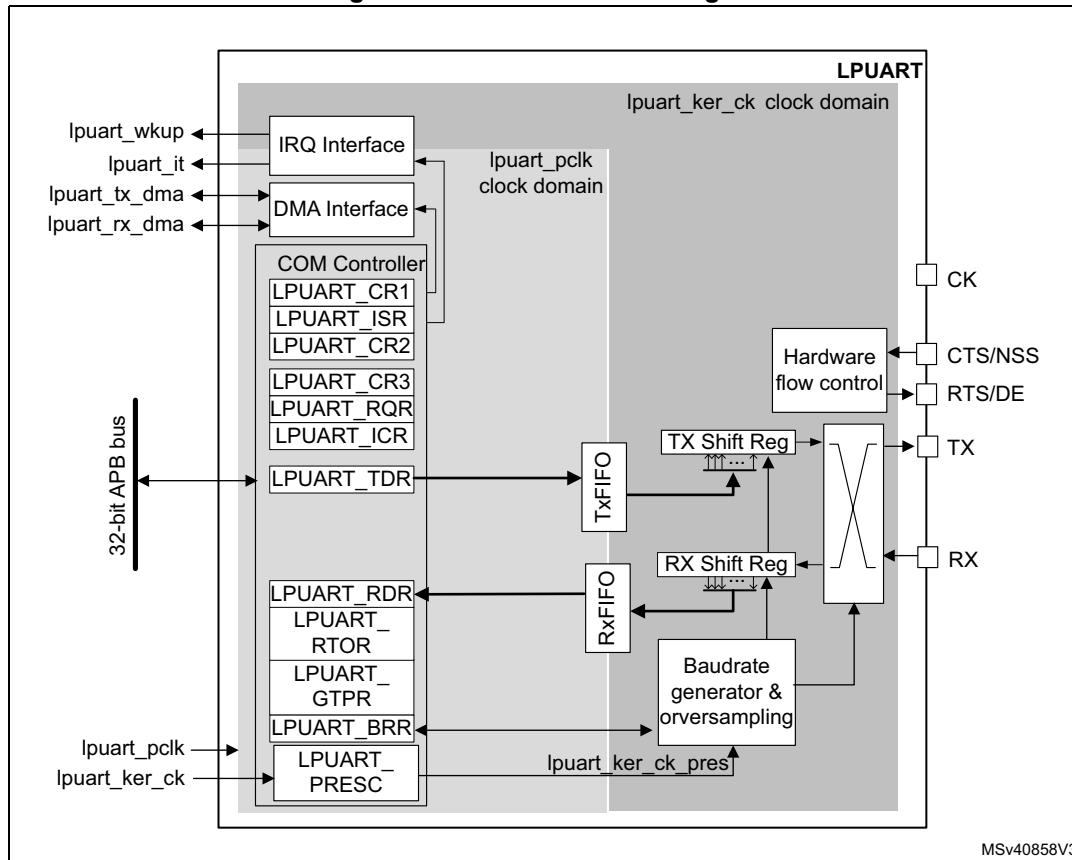
USART / LPUART modes/features <sup>(1)</sup>	USART1	LPUART1
Hardware flow control for modem	X	X
Continuous communication using DMA	X	X
Multiprocessor communication	X	X
Synchronous mode (Master/Slave)	X	-
Smartcard mode	X	-
Single-wire Half-duplex communication	X	X
IrDA SIR ENDEC block	X	-
LIN mode	X	-
Dual clock domain and wakeup from low-power mode	X	X
Receiver timeout interrupt	X	-
Modbus communication	X	-
Auto baud rate detection	X	-
Driver Enable	X	X
USART data length	7, 8 and 9 bits	
Tx/Rx FIFO	X	X
Tx/Rx FIFO size	8	

1. X = supported.

## 34.4 LPUART functional description

### 34.4.1 LPUART block diagram

Figure 347. LPUART block diagram



The simplified block diagram given in [Figure 347](#) shows two fully independent clock domains:

- The **lpuart\_pclk** clock domain  
The **lpuart\_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.
  - The **lpuart\_ker\_ck** kernel clock domain  
The **lpuart\_ker\_ck** is the LPUART clock source. It is independent of the **lpuart\_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart\_ker\_ck** is stopped.
- When the dual clock domain feature is disabled, the **lpuart\_ker\_ck** is the same as the **lpuart\_pclk** clock.

There is no constraint between **lpuart\_pclk** and **lpuart\_ker\_ck**: **lpuart\_ker\_ck** can be faster or slower than **lpuart\_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

### 34.4.2 LPUART signals

LPUART bidirectional communications requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)  
RX is the serial data input.
- **TX** (Transmit Data Output)  
When the transmitter is disabled, the output pin returns to its I/O port configuration.  
When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In Single-wire mode, this I/O is used to transmit and receive the data.

#### RS232 hardware flow control mode

The following pins are required in RS232 Hardware flow control mode:

- **CTS** (Clear To Send)  
When driven high, this signal blocks the data transmission at the end of the current transfer.
- **RTS** (Request to send)  
When it is low, this signal indicates that the USART is ready to receive data.

#### RS485 hardware flow control mode

The following pin is required in RS485 Hardware control mode:

- **DE** (Driver Enable)  
This signal activates the transmission mode of the external transceiver.

*Note:* DE and RTS share the same pin.

### 34.4.3 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART\_CR1 register (see [Figure 321](#)).

- 7-bit character length: M[1:0] = '10'
- 8-bit character length: M[1:0] = '00'
- 9-bit character length: M[1:0] = '01'

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

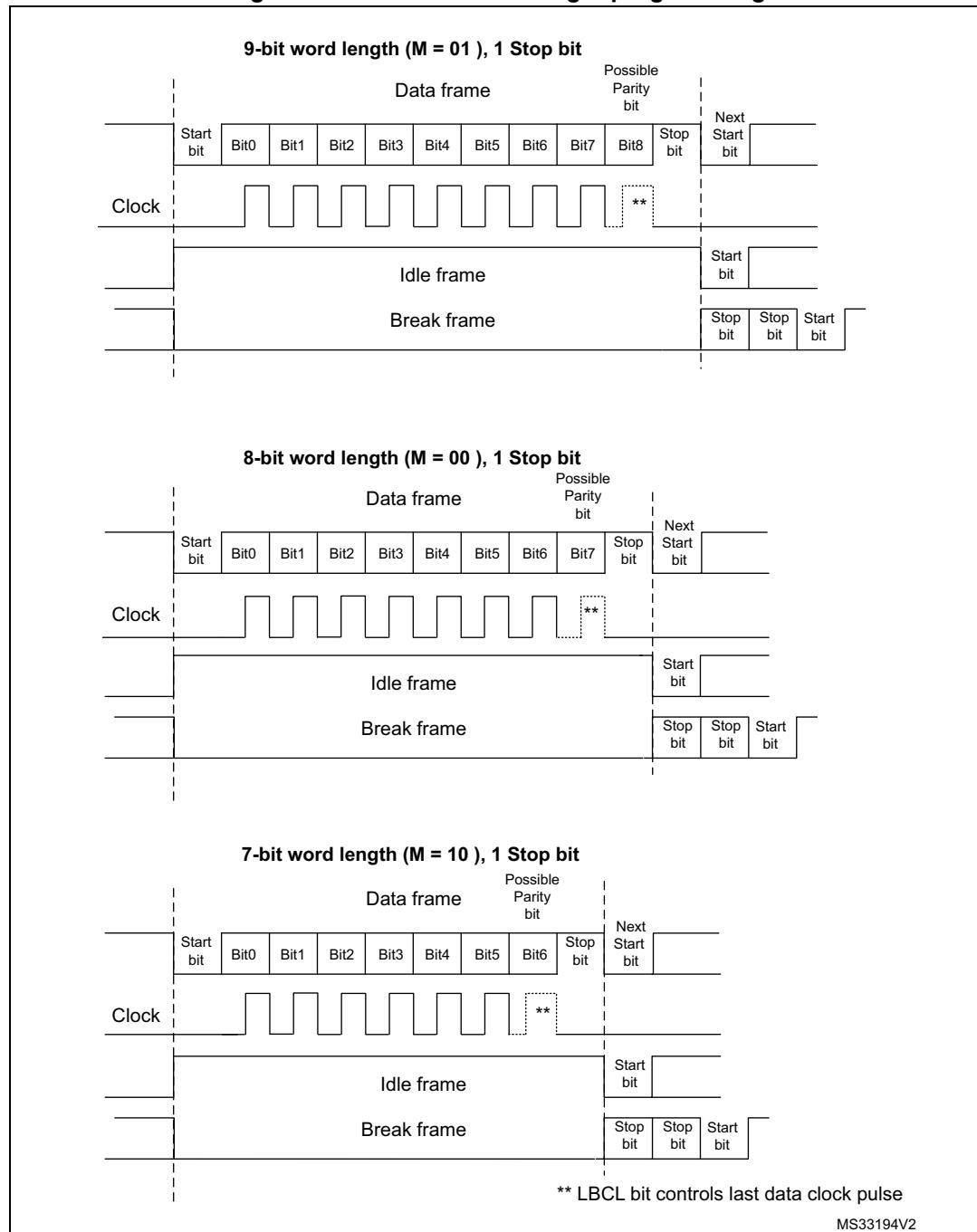
An **Idle character** is interpreted as an entire frame of "1"s. (The number of "1" 's includes the number of stop bits).

A **Break character** is interpreted on receiving "0"s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 348. LPUART word length programming



#### 34.4.4 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in LPUART\_CR1 register.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store

the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

**Note:** *The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.*

*The status flags are available in the LPUART\_ISR register.*

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in LPUART\_CR3 control register.

In this case:

- The RXFT flag is set in the LPUART\_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

RXFTCFG data have been received: one data in LPUART\_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to '101', the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size - 1 data in the RXFIFO and 1 data in the LPUART\_RDR. As a result, the next received data does not set the overrun flag.

- The TXFT flag is set in the LPUART\_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

### 34.4.5 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

#### Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART\_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see [Figure 347](#)).

When FIFO mode is enabled, the data written to the LPUART\_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

**Note:** *The TE bit must be set before writing the data to be transmitted to the LPUART\_TDR.*

*The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.*

*An idle frame is sent after the TE bit is enabled.*

### Configurable stop bits

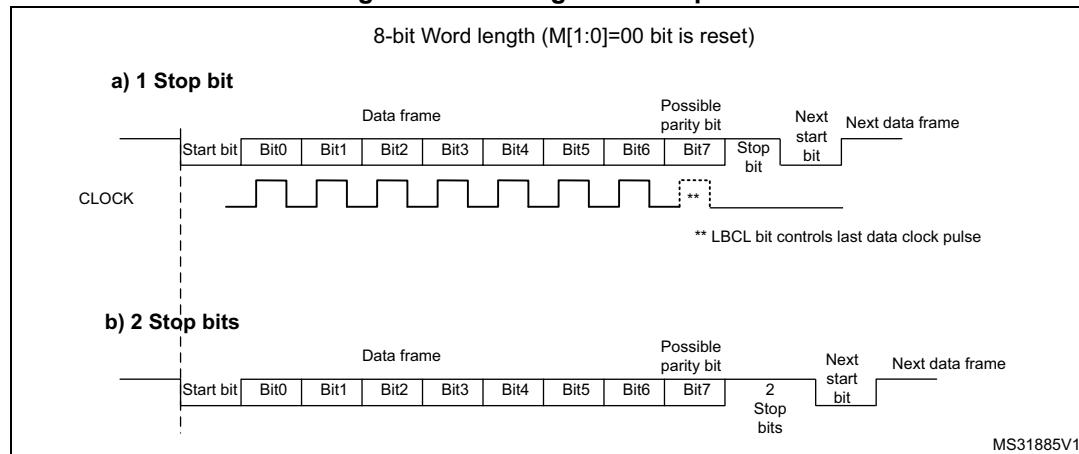
The number of stop bits to be transmitted with every character can be programmed in LPUART\_CR2 (bits 13,12).

- **1 stop bit:** This is the default value of number of stop bits.
- **2 Stop bits:** This is supported by normal LPUART, Single-wire and Modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = '00') or 11 low bits (when M[1:0] = '01') or 9 low bits (when M[1:0] = '10') followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 349. Configurable stop bits



### Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART\_CR1 to define the word length.
2. Select the desired baud rate using the LPUART\_BRR register.
3. Program the number of stop bits in LPUART\_CR2.
4. Enable the LPUART by writing the UE bit in LPUART\_CR1 register to '1'.
5. Select DMA enable (DMAT) in LPUART\_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in [Section 33.5.10: USART multiprocessor communication](#).
6. Set the TE bit in LPUART\_CR1 to send an idle frame as first transmission.
7. Write the data to send in the LPUART\_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
  - When FIFO mode is disabled, writing a data in the LPUART\_TDR clears the TXE flag.
  - When FIFO mode is enabled, writing a data in the LPUART\_TDR adds one data to the TXFIFO. Write operations to the LPUART\_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART\_TDR register, wait until TC = 1. This indicates that the transmission of the last frame is complete.
  - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.

- When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

### Single byte communication

- When FIFO mode disabled:

Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:

- the data have been moved from the LPUART\_TDR register to the shift register and data transmission has started;
- the LPUART\_TDR register is empty;
- the next data can be written to the LPUART\_TDR register without overwriting the previous data.

The TXE flag generates an interrupt if the TXEIE bit is set.

When a transmission is ongoing, a write instruction to the LPUART\_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

When no transmission is ongoing, a write instruction to the LPUART\_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO Not Full) flag is set by hardware to indicate that:

- the TXFIFO is not full;
- the LPUART\_TDR register is empty;
- the next data can be written to the LPUART\_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the LPUART\_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

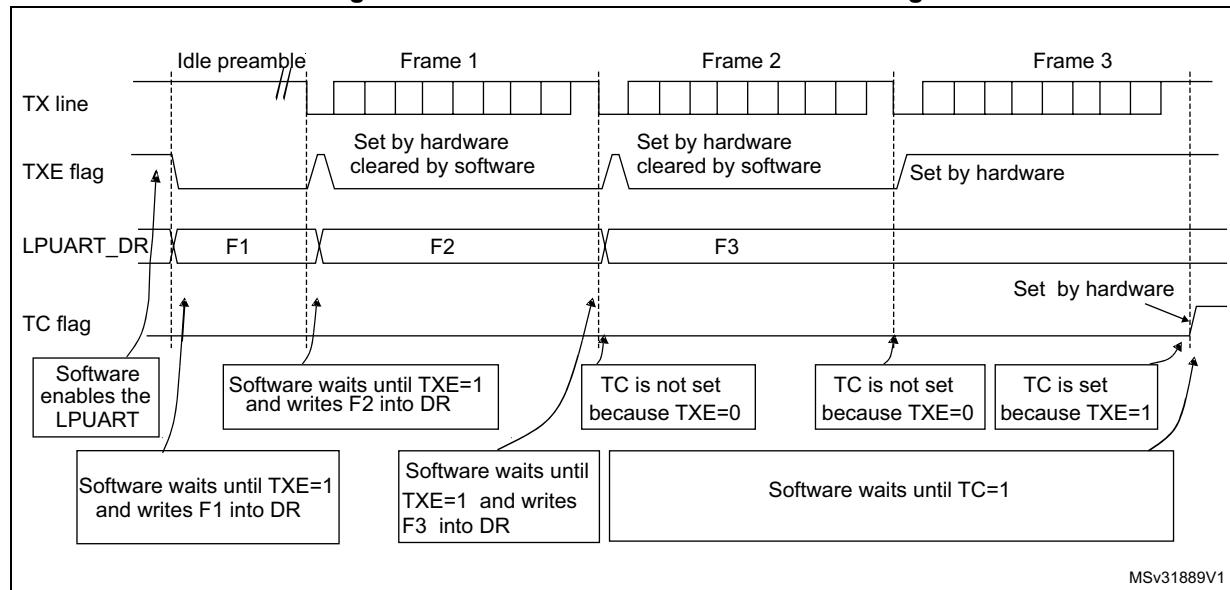
When the TXFIFO is not full, the TXFNF flag stays at '1' even after a write in LPUART\_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE is case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART\_CR1 register.

After writing the last data in the LPUART\_TDR register, it is mandatory to wait for TC = 1 before disabling the LPUART or causing the device to enter the low-power mode (see [Figure 350: TC/TXE behavior when transmitting](#)).

Figure 350. TC/TXE behavior when transmitting



**Note:** When FIFO management is enabled, the TXFNF flag is used for data transmission.

### Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see [Figure 348](#)).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

### Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

## 34.4.6 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART\_CR1 register.

### Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still '0'. If the start sample is at '1', then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

## Character reception

During an LPUART reception, data are shifted in least significant bit first (default configuration) through the RX pin. In this mode, the LPUART\_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

### Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART\_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART\_BRR
3. Program the number of stop bits in LPUART\_CR2.
4. Enable the LPUART by writing the UE bit in LPUART\_CR1 register to '1'.
5. Select DMA enable (DMAR) in LPUART\_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in [Section 33.5.10: USART multiprocessor communication](#).
6. Set the RE bit LPUART\_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART\_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the Receive Data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty i.e. there is a data in the RXFIFO to be read.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART\_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART\_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART\_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART\_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

### Break character

When a break character is received, the LPUART handles it as a framing error.

### Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

### Overrun error

- FIFO mode disabled

An overrun error occurs when a character is received when RXNE has not been reset.

Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

- the ORE bit is set;
- the RDR content is not lost. The previous data is available when a read to LPUART\_RDR is performed.;
- the shift register is overwritten. After that, any data received during overrun is lost.
- an interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

Data can not be transferred from the shift register to the LPUART\_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

- the ORE bit is set;
- the first entry in the RXFIFO is not lost. It is available when a read to LPUART\_RDR is performed.
- the shift register is overwritten. After that, any data received during overrun is lost.
- an interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

*Note:*

*The ORE bit, when set, indicates that at least 1 data has been lost. T*

*When the FIFO mode is disabled, there are two possibilities*

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

## Selecting the clock source

The choice of the clock source is done through the Clock Control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

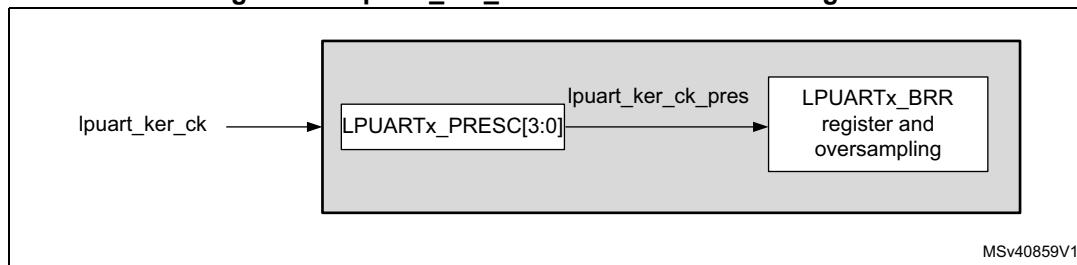
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is `Ipuart_ker_ck`.

When the dual clock domain and the wakeup from low-power mode features are supported, the `Ipuart_ker_ck` clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the `Ipuart_ker_ck` is the same as `Ipuart_pclk`.

The `Ipuart_ker_ck` can be divided by a programmable factor in the `LPUARTX_PRESC` register.

**Figure 351. `Ipuart_ker_ck` clock divider block diagram**



Some `Ipuart_ker_ck` sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the `LPUART_RDR` register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

*Note:*

*There is no noise detection for data.*

## Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the `LPUART_RDR` register.
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the `RXNE` bit which itself generates an interrupt. In case of

multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART\_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART\_ICR register.

### Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART\_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample i.e. during the second stop bit. The first stop bit is not checked for framing error.

### 34.4.7 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART\_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{Ipuartckpres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART\_BRR register.

**Note:** *The baud counters are updated to the new value in the baud registers after a write operation to LPUART\_BRR. Hence the baud rate register value should not be changed during communication.*

*It is forbidden to write values lower than 0x300 in the LPUART\_BRR register.*

*f<sub>CK</sub> must range from 3 x baud rate to 4096 x baud rate.*

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock. For example, if the LPUART clock source frequency is 100 MHz, the maximum baud rate that can be reached is about 33 Mbaud.

**Table 213. Error calculation for programmed baud rates at Ipuart\_ker\_ck\_pres = 32.768 kHz**

Baud rate		Ipuart_ker_ck_pres = 32.768 kHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	300 bauds	300 baud	0x6D3A	0
2	600 baud	600 baud	0x369D	0
3	1200 baud	1200.087 baud	0x1B4E	0.007
4	2400 baud	2400.17 baud	0xDA7	0.007
5	4800 baud	4801.72 baud	0x6D3	0.035
6	9600 baud	9608.94 baud	0x369	0.093

**Table 214. Error calculation for programmed baud rates at  $f_{CK} = 100$  MHz**

Baud rate		$f_{CK} = 100$ MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	38400 Baud	38400,04 Baud	A2C2A	0,0001
2	57600 Baud	57600,06 Baud	6C81C	0,0001
3	115200 Baud	115200,12 Baud	3640E	0,0001
4	230400 Baud	230400,23 Baud	1B207	0,0001
5	460800 Baud	460804,61 Baud	D903	0,001
6	921600 Baud	921625,81 Baud	6C81	0,0028
7	4000 Kbaud	4000000,00 Baud	1900	0
8	10000 Kbaud	10000000,00 Baud	A00	0
9	20000 Kbaud	20000000,00 Baud	500	0
10	33000 Kbaud	33032258,06 Baud	307	0,1

#### 34.4.8 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in [Table 215](#):

- Number of Stop bits defined through STOP[1:0] bits in the LPUART\_CR2 register
- LPUART\_BRR register value.

**Table 215. Tolerance of the LPUART receiver**

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M = '00'), 1 Stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M = '01'), 1 Stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M = '10'), 1 Stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M = '00'), 2 Stop bit	2.08%	2.86%	4.35%	4.42%
9 bits (M = '01'), 2 Stop bit	1.82%	2.56%	3.90%	4.42%
7 bits (M = '10'), 2 Stop bit	2.34%	3.23%	4.92%	4.42%

**Note:** The data specified in [Table 215](#) may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = '00' (11-bit times when M = '01' or 9-bit times when M = '10').

#### 34.4.9 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all non addressed receivers.

The non addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the LPUART\_CR1 register.

**Note:** When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart\_ker\_ck cycles), otherwise Mute mode might remain active.

When the Mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in LPUART\_ISR register is set to '1'. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART\_RQR register, under certain conditions.

The LPUART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the LPUART\_CR1 register:

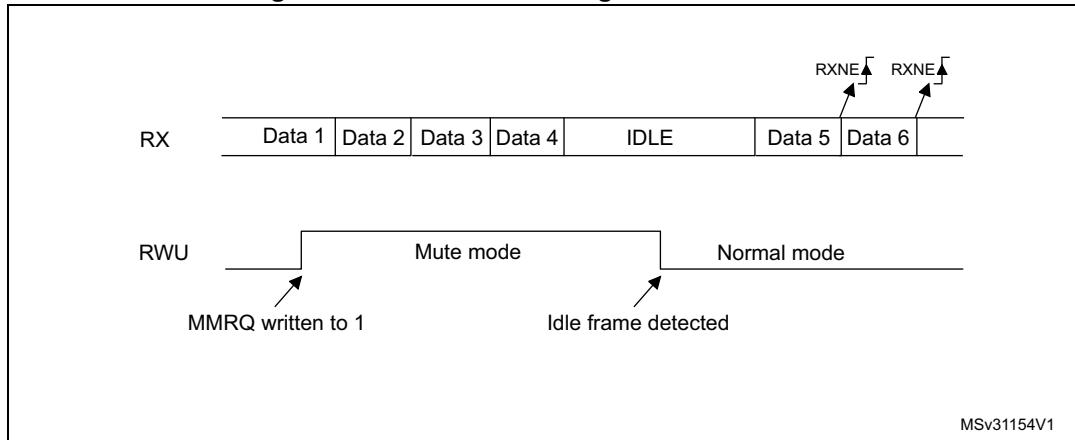
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

### Idle line detection (WAKE = 0)

The LPUART enters Mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART\_ISR register. An example of Mute mode behavior using Idle line detection is given in [Figure 352](#).

**Figure 352. Mute mode using Idle line detection**



**Note:** *If the MMRQ is set while the IDLE character has already elapsed, the Mute mode is not entered (RWU is not set).*

*If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).*

### 4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a '1' otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the LPUART\_CR2 register.

**Note:** *In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.*

The LPUART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters Mute mode.

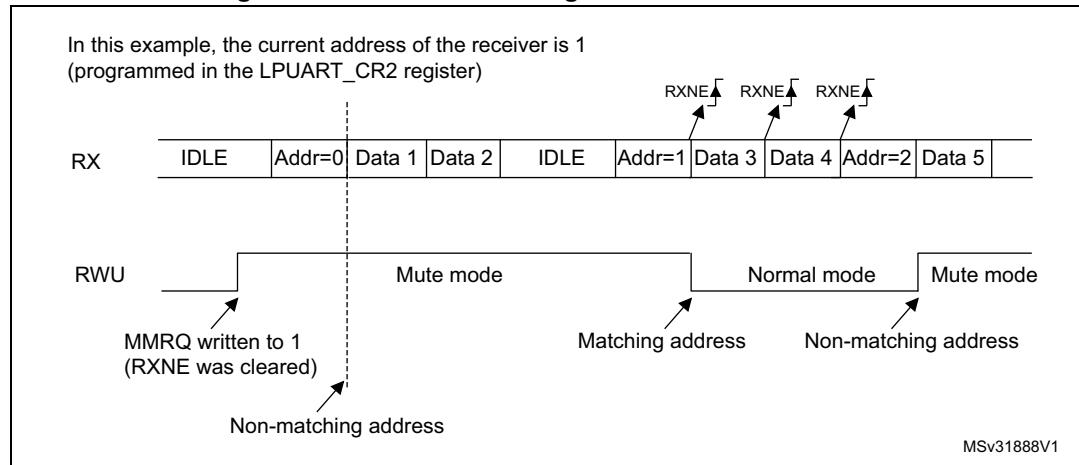
The LPUART also enters Mute mode when the MMRQ bit is written to '1'. The RWU bit is also automatically set in this case.

The LPUART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

**Note:** *When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in Mute mode.*

An example of Mute mode behavior using address mark detection is given in [Figure 353](#).

**Figure 353. Mute mode using address mark detection**



### 34.4.10 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART\_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in [Table 216](#).

**Table 216: LPUART frame formats**

M bits	PCE bit	LPUART frame <sup>(1)</sup>
00	0	SB   8 bit data   STB
00	1	SB   7-bit data   PB   STB
01	0	SB   9-bit data   STB
01	1	SB   8-bit data PB   STB
10	0	SB   7bit data   STB
10	1	SB   6-bit data   PB   STB

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

#### Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data equal 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART\_CR1 = 0).

#### Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data equal 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART\_CR1 = 1).

### Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART\_ISR register and an interrupt is generated if PEIE is set in the LPUART\_CR1 register. The PE flag is cleared by software writing 1 to the PECE in the LPUART\_ICR register.

### Parity generation in transmission

If the PCE bit is set in LPUART\_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

#### 34.4.11 LPUART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the LPUART\_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the LPUART\_CR2 register,
- SCEN and IREN bits in the LPUART\_CR3 register.

The LPUART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in LPUART\_CR3.

As soon as HDSEL is written to ‘1’:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

**Note:** *In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.*

#### 34.4.12 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

**Note:** *Refer to [Section 33.4: USART implementation on page 1038](#) to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in [Section 33.5.6](#). To perform continuous communication. When FIFO is disabled, you can clear the TXE/ RXNE flags in the LPUART\_ISR register.*

### Transmission using DMA

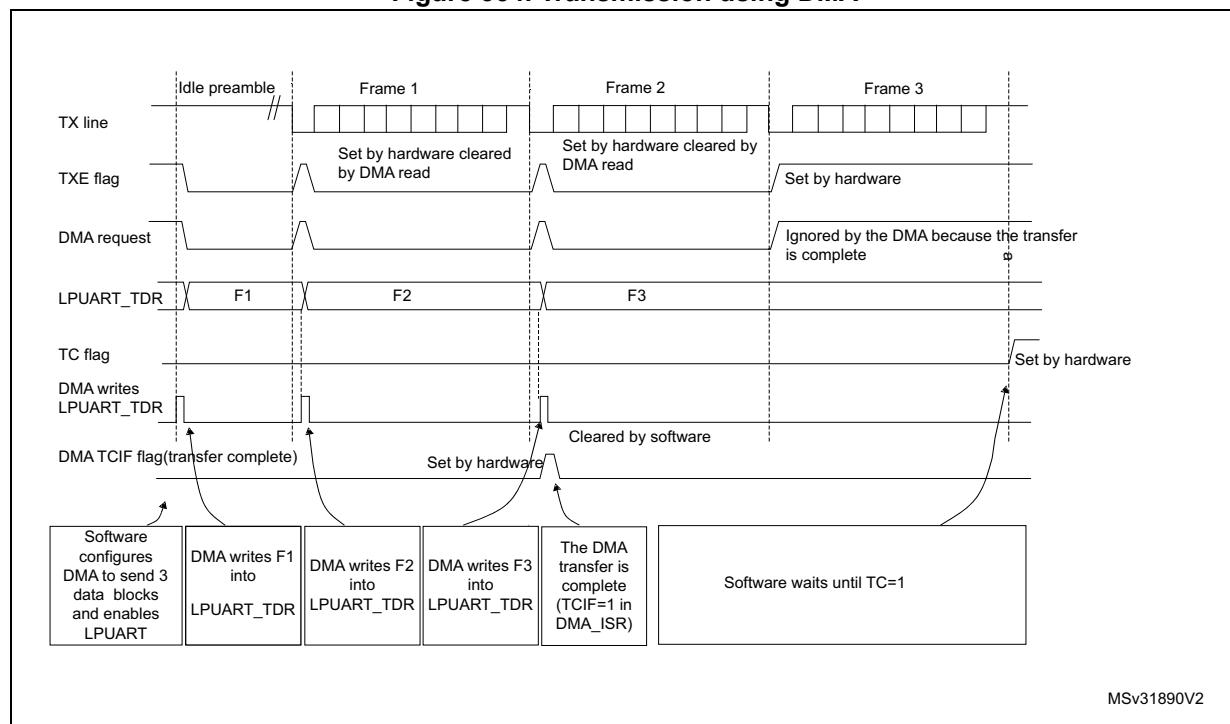
DMA mode can be enabled for transmission by setting DMAT bit in the LPUART\_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding [Direct memory access controller section](#)) to the LPUART\_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

1. Write the LPUART\_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART\_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART\_ISR register by setting the TCCF bit in the LPUART\_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA\_ISR register), the TC flag can be monitored to make sure that the LPUART communication is complete. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 354. Transmission using DMA



**Note:** When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

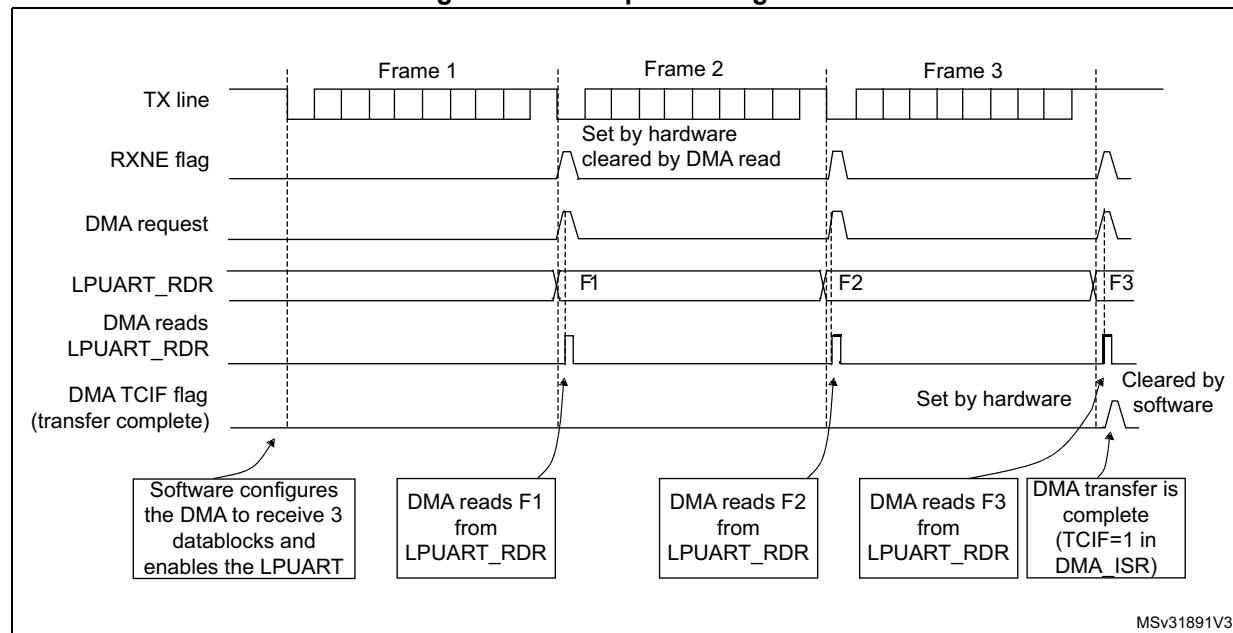
### Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in LPUART\_CR3 register. Data are loaded from the LPUART\_RDR register to a SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller (DMA)* section) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART\_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART\_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

**Figure 355. Reception using DMA**



**Note:** When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

### Error flagging and interrupt generation in multibuffer communication

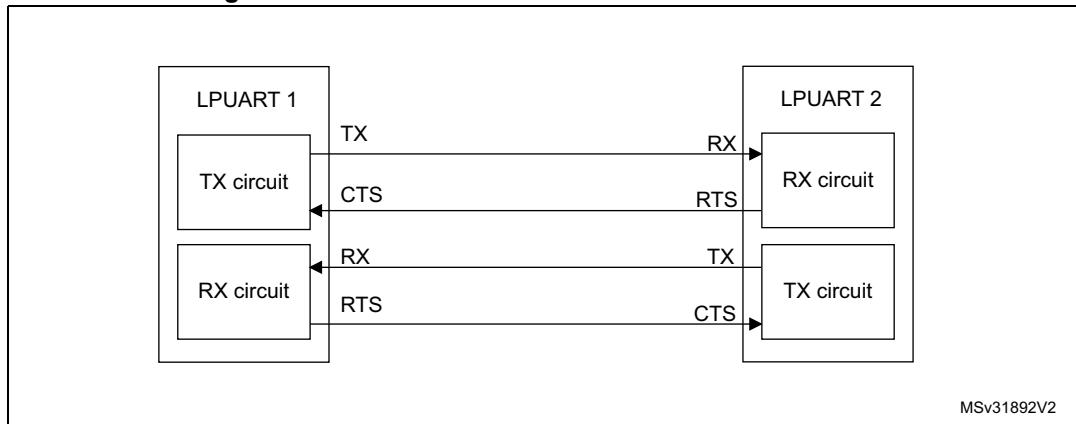
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt

enable bit (EIE bit in the LPUART\_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

### 34.4.13 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The [Figure 342](#) shows how to connect 2 devices in this mode:

**Figure 356. Hardware flow control between 2 LPUARTs**

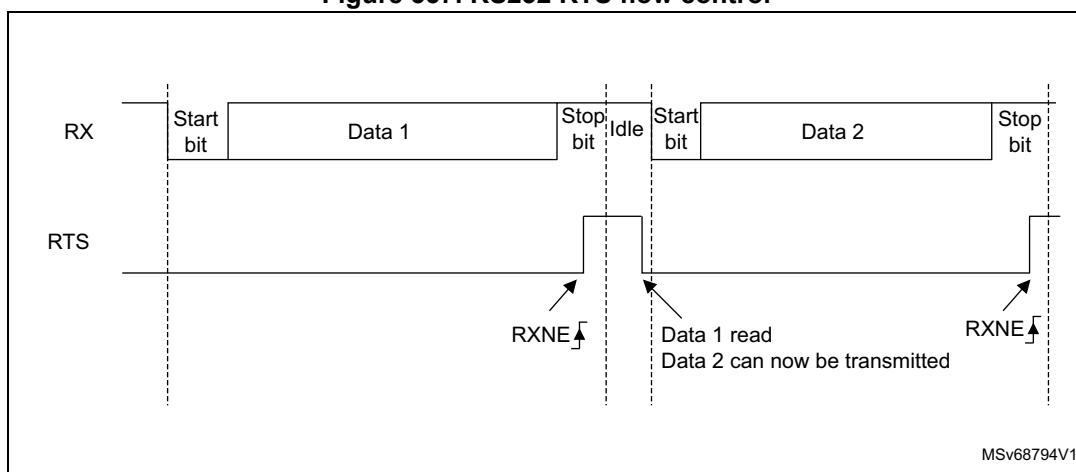


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART\_CR3 register).

#### RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. [Figure 357](#) shows an example of communication with RTS flow control enabled.

**Figure 357. RS232 RTS flow control**



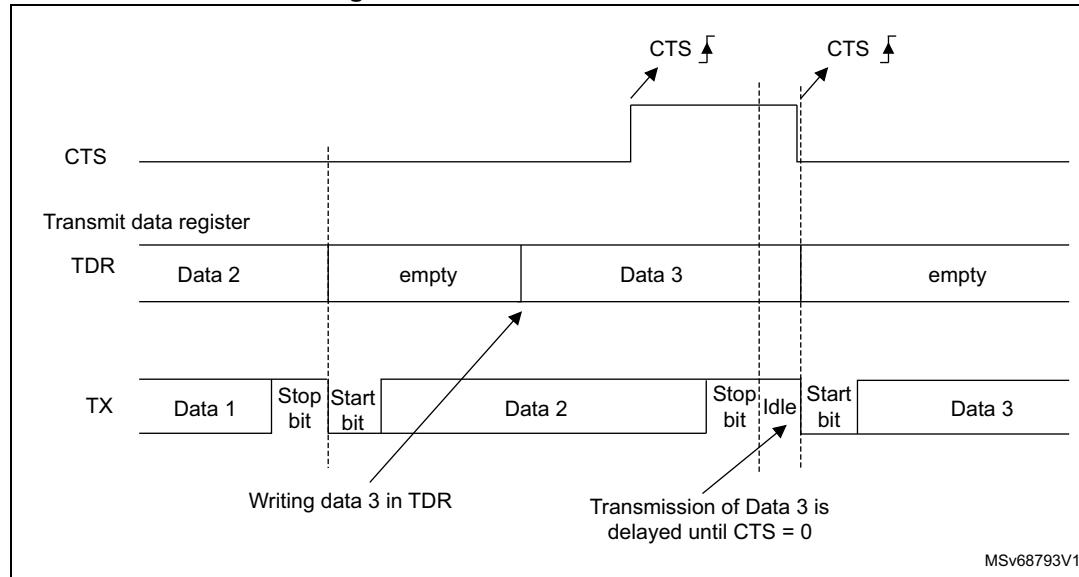
**Note:** When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

### RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART\_CR3 register is set. [Figure 358](#) shows an example of communication with CTS flow control enabled.

**Figure 358. RS232 CTS flow control**



**Note:**

*For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.*

### RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART\_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART\_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART\_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART\_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source ( $f_{CK}$ ) cycles:

- The Driver enable assertion time equals
  - $(1 + (DEAT \times P)) \times f_{CK}$ , if  $P \neq 0$
  - $(1 + DEAT) \times f_{CK}$ , if  $P = 0$
- The Driver enable deassertion time equals
  - $(1 + (DEDT \times P)) \times f_{CK}$ , if  $P \neq 0$
  - $(1 + DEDT) \times f_{CK}$ , if  $P = 0$

where  $P = BRR[20:11]$

#### 34.4.14 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the Ipuart\_pclk clock is disabled.

The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set. When the Ipuart\_pclk is gated, the LPUART provides a wakeup interrupt (Ipuart\_wkup) if a specific action requiring the activation of the Ipuart\_pclk clock is needed:

- If FIFO mode is disabled
 

Iuart\_pclk clock has to be activated to empty the LPUART data register.

In this case, the Ipuart\_wkup interrupt source is the RXNE set to '1'. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled
 

Iuart\_pclk clock has to be activated

  - to fill the TXFIFO
  - or to empty the RXFIFO

In this case, the Ipuart\_wkup interrupt source can be:

  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

To avoid overrun/underrun errors and transmit/receive data in low-power mode, the Ipuart\_wkup interrupt source can be one of the following events:

- TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
- RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from low-power mode enables doing as many LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific lpuart\_wkup interrupt may be selected through the WUS bitfields.

When the wakeup event is detected, the WUF flag is set by hardware and lpuart\_wkup interrupt is generated if the WUFIE bit is set.

**Note:** *Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.*

*The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in low-power or in an active mode.*

*When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the LPUART is actually enabled.*

*When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled upon exit from low-power mode.*

*When FIFO is enabled, the wakeup from low-power mode on address match is only possible when Mute mode is enabled.*

### Using Mute mode with low-power mode

If the LPUART is put into Mute mode before entering low-power mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wakeup from Mute mode on address match is used, then the low-power mode wakeup source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in Mute mode upon address match and wake up from low-power mode.

**Note:** *When FIFO management is enabled, Mute mode is used with wakeup from low-power mode without any constraints (i.e. the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).*

### Wakeup from low-power mode when LPUART kernel clock lpuart\_ker\_ck is OFF in low-power mode

If during low-power mode, the lpuart\_ker\_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart\_ker\_ck clock to be switched ON thanks to the lpuart\_ker\_ck\_req signal. The lpuart\_ker\_ck is then used for the frame reception.

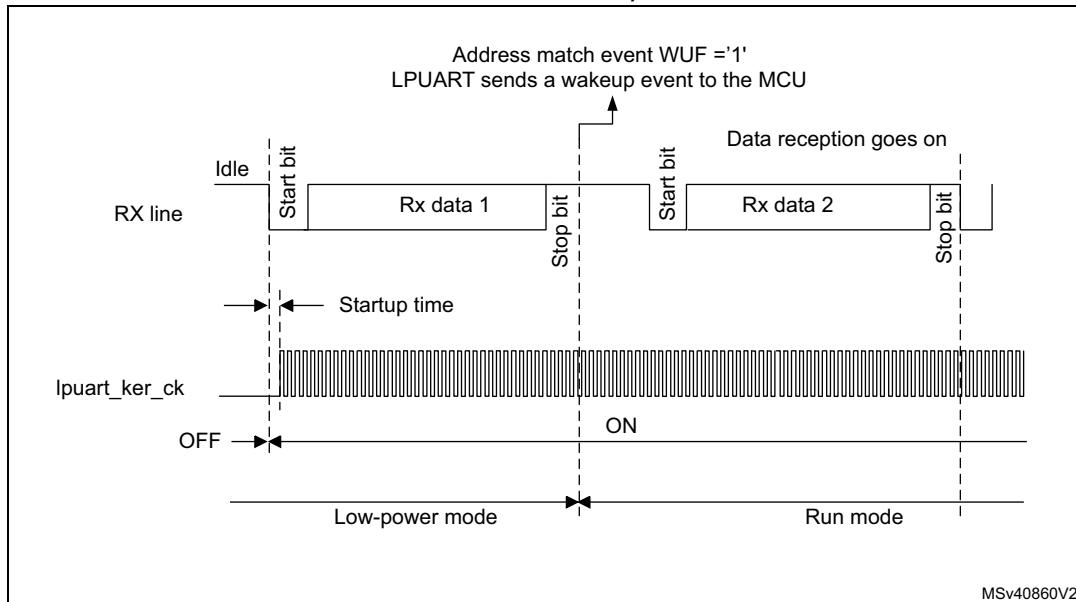
If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wakeup event is not verified, the lpuart\_ker\_ck is switched OFF again, the MCU is not waken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of wakeup event programmed to “address match detection” and FIFO management disabled.

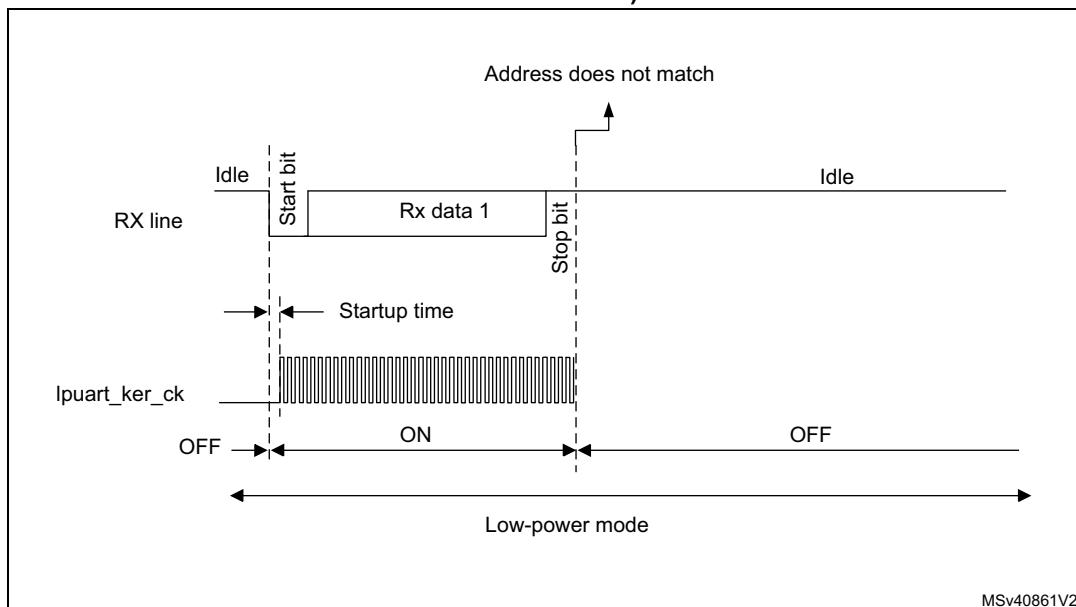
*Figure 359* shows the behavior when the wakeup event is verified.

**Figure 359. Wakeup event verified (wakeup event = address match, FIFO disabled)**



[Figure 360](#) shows the behavior when the wakeup event is not verified.

**Figure 360. Wakeup event not verified (wakeup event = address match, FIFO disabled)**



Note:

The above figures are valid when address match or any received frame is used as wakeup event. In the case the wakeup event is the start bit detection, the LPUART sends the wakeup event to the MCU at the end of the start bit.

### Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see [Section 34.4.8: Tolerance of the LPUART receiver to clock deviation](#)).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to [Table 215: Tolerance of the LPUART receiver](#), the LPUART receiver tolerance equals 3.41%.

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

$$D_{WU\max} = t_{WULPUART} / (11 \times T_{bit\ Min})$$

$$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WU\max})$$

where  $t_{WULPUART}$  is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart\_ker\_ck inaccuracy.

For example, if HSI is used as lpuart\_ker\_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART} = 3 \mu\text{s}$  (values provided only as examples; for correct values, refer to the device datasheet).

$$D_{WU\max} = 3.41\% - 1\% = 2.41\%$$

$$T_{bit\ min} = 3 \mu\text{s} / (11 \times 2.41\%) = 11.32 \mu\text{s}.$$

As a result, the maximum baud rate that enables to wakeup correctly from low-power mode is:  $1/11.32 \mu\text{s} = 88.36 \text{ kbaud}$ .

## 34.5 LPUART in low-power modes

Table 217. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop <sup>(1)</sup>	The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode.
Standby	The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to [Section 34.3: LPUART implementation](#) to know if the wakeup from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

## 34.6 LPUART interrupts

Refer to [Table 218](#) for a detailed description of all LPUART interrupt requests.

**Table 218. LPUART interrupt requests**

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop <sup>(1)</sup> modes	Exit from Standby mode
LPUART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF <sup>(2)</sup>	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX-FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF	Yes	No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE <sup>(3)</sup>		Write 1 in ORECF		No	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Wakeup from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the Wakeup from Stop mode feature. Refer to [Section 34.3: LPUART implementation](#) for the list of supported Stop modes.
2. RXFF flag is asserted if the LPUART receives  $n+1$  data ( $n$  being the RXFIFO size):  $n$  data in the RXFIFO and 1 data in LPUART\_RDR. In Stop mode, LPUART\_RDR is not clocked. As a result, this register is not written and once  $n$  data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

## 34.7 LPUART registers

Refer to [Section 1.2 on page 61](#) for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

### 34.7.1 LPUART control register 1 (LPUART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

#### FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXF FIE	TXFEIE	FIFO EN	M1	Res.	Res.	DEAT[4:0]								DEDT[4:0]	
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFN FIE	TCIE	RXFN EIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

#### Bit 31 **RXFFIE**:RXFIFO full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated when RXFF = 1 in the LPUART\_ISR register

#### Bit 30 **TXFEIE**:TXFIFO empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated when TXFE = 1 in the LPUART\_ISR register

#### Bit 29 **FIFOEN**:FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = '00': 1 Start bit, 8 Data bits, n Stop bit

M[1:0] = '01': 1 Start bit, 9 Data bits, n Stop bit

M[1:0] = '10': 1 Start bit, 7 Data bits, n Stop bit

This bit can only be written when the LPUART is disabled (UE = 0).

*Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.*

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart\_ker\_ck clock cycles. For more details, refer [Section 33.5.20: RS232 Hardware flow control and RS485 Driver Enable](#).

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in lpuart\_ker\_ck clock cycles. For more details, refer [Section 34.4.13: RS232 Hardware flow control and RS485 Driver Enable](#).

If the LPUART\_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART\_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if  $M = 1$ ; 8th bit if  $M = 0$ ) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

- 0: Parity control disabled
- 1: Parity control enabled

This bitfield can only be written when the LPUART is disabled ( $UE = 0$ ).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

- 0: Even parity
- 1: Odd parity

This bitfield can only be written when the LPUART is disabled ( $UE = 0$ ).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever  $PE = 1$  in the LPUART\_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated whenever  $TXE/TXFNF = 1$  in the LPUART\_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever  $TC = 1$  in the LPUART\_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated whenever  $ORE = 1$  or  $RXNE/RXFNE = 1$  in the LPUART\_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever  $IDLE = 1$  in the LPUART\_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

- 0: Transmitter is disabled
- 1: Transmitter is enabled

*Note: During transmission, a low pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART\_ISR register.*

*When TE is set there is a 1 bit-time delay before the transmission starts.*

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in Stop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.

When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

*Note: It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.*

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART\_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART\_ISR to be set before resetting the UE bit.*

*The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.*

### 34.7.2 LPUART control register 1 [alternate] (LPUART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

#### FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]			
		rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = '00': 1 Start bit, 8 Data bits, n Stop bit

M[1:0] = '01': 1 Start bit, 9 Data bits, n Stop bit

M[1:0] = '10': 1 Start bit, 7 Data bits, n Stop bit

This bit can only be written when the LPUART is disabled (UE = 0).

*Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.*

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in Ipuart\_ker\_ck clock cycles. For more details, refer [Section 33.5.20: RS232 Hardware flow control and RS485 Driver Enable](#).

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in Ipuart\_ker\_ck clock cycles. For more details, refer [Section 34.4.13: RS232 Hardware flow control and RS485 Driver Enable](#).

If the LPUART\_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART\_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART\_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART\_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART\_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE/RXFNE = 1 in the LPUART\_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART\_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

*Note: During transmission, a low pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART\_ISR register.*

*When TE is set there is a 1 bit-time delay before the transmission starts.*

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in Stop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.

When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

*Note: It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.*

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART\_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART\_ISR to be set before resetting the UE bit.*

*The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.*

### 34.7.3 LPUART control register 2 (LPUART\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.	Res.	Res.	Res.	MSBF1 RST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	ADDM7	Res.	Res.	Res.	Res.						
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in Mute mode or a character code to be recognized in low-power or Run mode:

- In Mute mode: they are used in multiprocessor communication to wakeup from Mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wakeup from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with Mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the USART is disabled (UE = 0).

## Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV**: Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H).

The parity bit is also inverted.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV**: TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels ( $V_{DD} = 1/\text{idle}$ , Gnd = 0/mark)

1: TX pin signal values are inverted ( $V_{DD} = 0/\text{mark}$ , Gnd = 1/idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 16 **RXINV**: RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels ( $V_{DD} = 1/\text{idle}$ , Gnd = 0/mark)

1: RX pin signal values are inverted ( $V_{DD} = 0/\text{mark}$ , Gnd = 1/idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 **SWAP**: Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the LPUART is disabled (UE = 0).

## Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: STOP bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: Reserved.

10: 2 stop bits

11: Reserved

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7:7**-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the LPUART is disabled (UE = 0)

*Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.*

Bits 3:0 Reserved, must be kept at reset value.

**34.7.4 LPUART control register 3 (LPUART\_CR3)**

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS[1:0]		Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG[2:0]**: TXFIFO threshold configuration

- 000:TXFIFO reaches 1/8 of its depth.
  - 001:TXFIFO reaches 1/4 of its depth.
  - 110:TXFIFO reaches 1/2 of its depth.
  - 011:TXFIFO reaches 3/4 of its depth.
  - 100:TXFIFO reaches 7/8 of its depth.
  - 101:TXFIFO becomes empty.
- Remaining combinations: Reserved.

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt is inhibited
  - 1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration

- 000:Receive FIFO reaches 1/8 of its depth.
  - 001:Receive FIFO reaches 1/4 of its depth.
  - 110:Receive FIFO reaches 1/2 of its depth.
  - 011:Receive FIFO reaches 3/4 of its depth.
  - 100:Receive FIFO reaches 7/8 of its depth.
  - 101:Receive FIFO becomes full.
- Remaining combinations: Reserved.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt is inhibited
  - 1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from low-power mode interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt is inhibited
  - 1: An LPUART interrupt is generated whenever WUF = 1 in the LPUART\_ISR register

*Note: WUFIE must be set before entering in low-power mode.*

*If the LPUART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation.*

Bits 21:20 **WUS[1:0]**: Wakeup from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (Wakeup from low-power mode flag).

- 00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
- 01: Reserved.
- 10: WUF active on Start bit detection
- 11: WUF active on RXNE.

This bitfield can only be written when the LPUART is disabled (UE = 0).

*Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation.*

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

- 0: DE signal is active high.
- 1: DE signal is active low.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

- 0: DE function is disabled.
- 1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA disable on reception error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the LPUART is disabled (UE = 0).

*Note: The reception errors are: parity error, framing error or noise error.*

Bit 12 **OVRDIS**: Overrun disable

This bit is used to disable the receive overrun detection.

- 0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
- 1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART\_RDR register.

This bit can only be written when the LPUART is disabled (UE = 0).

*Note: This control bit enables checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable

- 0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the LPUART\_ISR register

Bit 9 **CTSE**: CTS enable

- 0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable

- 0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

- 1: DMA mode is enabled for transmission

- 0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART\_ISR register).

0: Interrupt is inhibited

1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART\_ISR register.

### 34.7.5 LPUART baud rate register (LPUART\_BRR)

This register can only be written when the LPUART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]**: LPUART baud rate

*Note:* It is forbidden to write values lower than 0x300 in the LPUART\_BRR register.

*Provided that LPUART\_BRR must be  $\geq 0x300$  and LPUART\_BRR is 20 bits, a care should be taken when generating high baud rates using high fck values. fck must be in the range [3 x baud rate..4096 x baud rate].*

### 34.7.6 LPUART request register (LPUART\_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.										
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART\_ISR register).

*Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.*

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

*Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.*

Bit 0 Reserved, must be kept at reset value.

### 34.7.7 LPUART interrupt and status register (LPUART\_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

#### FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG in LPUART\_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART\_CR3 register.

0: TXFIFO does not reach the programmed threshold.  
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the threshold programmed in RXFTCFG in LPUART\_CR3 register i.e. the Receive FIFO contains RXFTCFG data. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the LPUART\_CR3 register.

0: Receive FIFO does not reach the programmed threshold.  
1: Receive FIFO reached the programmed threshold.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART\_RDR register).

An interrupt is generated if the RXFFIE bit = 1 in the LPUART\_CR1 register.  
0: RXFIFO is not full  
1: RXFIFO is full

Bit 23 **TXFE**: TXFIFO empty

This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART\_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART\_CR1 register.  
0: TXFIFO is not empty  
1: TXFIFO is empty

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

*Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.*

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART\_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART\_ICR register. An interrupt is generated if WUFIE = 1 in the LPUART\_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*

*If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value*

Bit 19 **RWU**: Receiver wakeup from Mute mode

This bit indicates if the LPUART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART\_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART\_RQR register.

- 0: Receiver in Active mode
- 1: Receiver in Mute mode

*Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.*

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART\_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

- 0: Break character transmitted
- 1: Break character requested by setting SBKRQ bit in LPUART\_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART\_ICR register.

An interrupt is generated if CMIE = 1 in the LPUART\_CR1 register.

- 0: No Character match detected
- 1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

- 0: LPUART is idle (no reception)
- 1: Reception on going

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

- 0: CTS line set
- 1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART\_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART\_CR3 register.

- 0: No change occurred on the CTS status line
- 1: A change occurred on the CTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXFNF**: TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART\_TDR. Every write in the LPUART\_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART\_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART\_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

*Note: This bit is used during single buffer transmission.*

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXFF is set. An interrupt is generated if TCIE = 1 in the LPUART\_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART\_ICR register or by a write to the LPUART\_TDR register.

An interrupt is generated if TCIE = 1 in the LPUART\_CR1 register.

0: Transmission is not complete

1: Transmission is complete

*Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.*

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART\_RDR register. Every read of the LPUART\_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART\_RQR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART\_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART\_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART\_ICR register.

0: No Idle line is detected

1: Idle line is detected

*Note: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).*

*If Mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the Mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.*

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART\_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART\_ICR register.

An interrupt is generated if RXFNEIE = 1 or EIE = 1 in the LPUART\_CR1 register, or EIE = 1 in the LPUART\_CR3 register.

0: No overrun error

1: Overrun error is detected

*Note: When this bit is set, the LPUART\_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

*This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART\_CR3 register.*

Bit 2 **NE**: Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NECF bit in the LPUART\_ICR register.

0: No noise is detected

1: Noise is detected

*Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.*

*This error is associated with the character in the LPUART\_RDR.*

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART\_ICR register. When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART\_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

*Note: This error is associated with the character in the LPUART\_RDR.*

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART\_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART\_CR1 register.

0: No parity error

1: Parity error

*Note: This error is associated with the character in the LPUART\_RDR.*

### 34.7.8 LPUART interrupt and status register [alternate] (LPUART\_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

#### FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

#### Bit 22 REACK: Receive enable acknowledge flag

This bit is set/reset by hardware when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

*Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.*

#### Bit 21 TEACK: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART\_CR1 register, in order to respect the TE = 0 minimum period.

#### Bit 20 WUF: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART\_ICR register. An interrupt is generated if WUFIE = 1 in the LPUART\_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*

*If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value*

#### Bit 19 RWU: Receiver wakeup from Mute mode

This bit indicates if the LPUART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART\_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART\_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

*Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.*

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART\_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: Break character transmitted

1: Break character requested by setting SBKRQ bit in LPUART\_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART\_ICR register.

An interrupt is generated if CMIE = 1 in the LPUART\_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception on going

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART\_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART\_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE**: Transmit data register empty/TXFIFO not full

TXE is set by hardware when the content of the LPUART\_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART\_TDR register.

An interrupt is generated if the TXEIE bit =1 in the LPUART\_CR1 register.

0: Data register full

1: Data register not full

*Note: This bit is used during single buffer transmission.*

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE = 1 in the LPUART\_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART\_ICR register or by a write to the LPUART\_TDR register.

An interrupt is generated if TCIE = 1 in the LPUART\_CR1 register.

0: Transmission is not complete

1: Transmission is complete

*Note: If TE bit is reset and no transmission is on going, the TC bit is immediately set.*

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART\_RDR shift register has been transferred to the LPUART\_RDR register. It is cleared by reading from the LPUART\_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART\_RQR register.

An interrupt is generated if RXNEIE = 1 in the LPUART\_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART\_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART\_ICR register.

0: No Idle line is detected

1: Idle line is detected

*Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).*

*If Mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the Mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.*

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART\_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART\_ICR register.

An interrupt is generated if RXNEIE = 1 or EIE = 1 in the LPUART\_CR1 register, or EIE = 1 in the LPUART\_CR3 register.

0: No overrun error

1: Overrun error is detected

*Note: When this bit is set, the LPUART\_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

*This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART\_CR3 register.*

Bit 2 **NE**: Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NECF bit in the LPUART\_ICR register.

0: No noise is detected

1: Noise is detected

*Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.*

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART\_ICR register. When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART\_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART\_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART\_CR1 register.

0: No parity error

1: Parity error

### 34.7.9 LPUART interrupt flag clear register (LPUART\_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.						
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLEC F	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the LPUART\_ISR register.

*Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation.*

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART\_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART\_ISR register.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the LPUART\_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag

Writing 1 to this bit clears the IDLE flag in the LPUART\_ISR register.

Bit 3 **ORECF**: Overrun error clear flag

Writing 1 to this bit clears the ORE flag in the LPUART\_ISR register.

Bit 2 **NECF**: Noise detected clear flag

Writing 1 to this bit clears the NE flag in the LPUART\_ISR register.

Bit 1 **FECF**: Framing error clear flag

Writing 1 to this bit clears the FE flag in the LPUART\_ISR register.

Bit 0 **PECF**: Parity error clear flag

Writing 1 to this bit clears the PE flag in the LPUART\_ISR register.

### 34.7.10 LPUART receive data register (LPUART\_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RDR[8:0]														
								r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see [Figure 347](#)).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

### 34.7.11 LPUART transmit data register (LPUART\_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDR[8:0]														
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see [Figure 347](#)).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART\_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

*Note: This register must be written only when TXE/TXFNF = 1.*

### 34.7.12 LPUART prescaler register (LPUART\_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
PRESCALER[3:0]															
														rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided

0001: input clock divided by 2

0010: input clock divided by 4

0011: input clock divided by 6

0100: input clock divided by 8

0101: input clock divided by 10

0110: input clock divided by 12

0111: input clock divided by 16

1000: input clock divided by 32

1001: input clock divided by 64

1010: input clock divided by 128

1011: input clock divided by 256

Remaining combinations: Reserved.

*Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is 1011 i.e. input clock divided by 256.*

### 34.7.13 LPUART register map

The table below gives the LPUART register map and reset values.

Table 219. LPUART register map and reset values

Offset	Register name	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x00	LPUART_CR1 FIFO mode enabled	Res.	RXFFIE	0	TXFFIE	0																															
	Reset value	0	0	0	0	0	M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x00	LPUART_CR1 FIFO mode disabled	Res.	Res.	0	FIFOEN	0	FIFOEN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x04	LPUART_CR2																																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x08	LPUART_CR3																																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																																				
0x10-0x14																																					
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																																				
0x1C	LPUART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																																				
0x1C	LPUART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																																				
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																																				
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																																				
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																				

Table 219. LPUART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	LPUART_PRESC	Res.	PRESCALER[3:0]																														
	Reset value																												0	0	0	0	0

Refer to [Section 2.2: Memory organization](#) for the register boundary addresses.

## 35 Serial peripheral interface (SPI)

### 35.1 Introduction

The SPI interface can be used to communicate with external devices using the SPI protocol. SPI mode is selectable by software. SPI Motorola mode is selected by default after a device reset.

The serial peripheral interface (SPI) protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

### 35.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4 to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to  $f_{PCLK}/2$
- Slave mode frequency up to  $f_{PCLK}/2$ .
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- Enhanced TI and NSS pulse modes support

### 35.3 SPI implementation

The following table describes all the SPI instances and their features embedded in the devices.

Table 220. SPI implementation

SPI Features	SPI1	SPI2 <sup>(1)</sup>
Enhanced NSSP & TI modes	Yes	Yes
I <sup>2</sup> S support	No	No
Hardware CRC calculation	Yes	Yes
Data size configurable	from 4 to 16-bit	from 4 to 16-bit
Rx/Tx FIFO size	32-bit	32-bit
Wakeup capability from Low-power Sleep	Yes	Yes

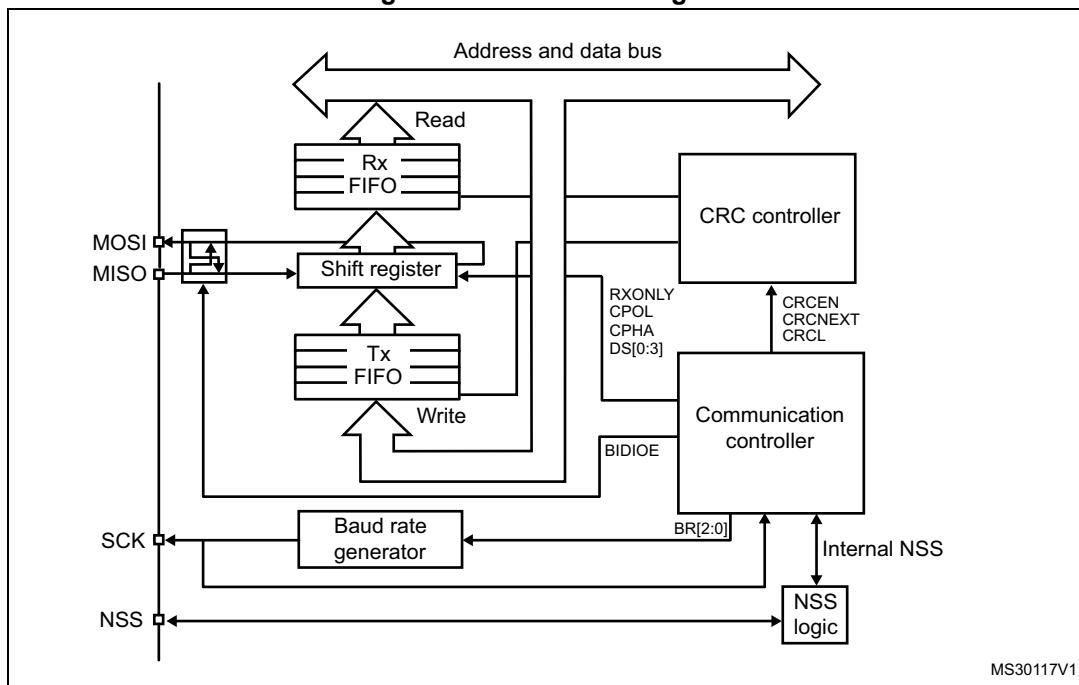
1. Available only on STM32WB55xx devices.

## 35.4 SPI functional description

### 35.4.1 General description

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram [Figure 361](#).

Figure 361. SPI block diagram



Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame or
  - detect a conflict between multiple masters

See [Section 35.4.5: Slave select \(NSS\) pin management](#) for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

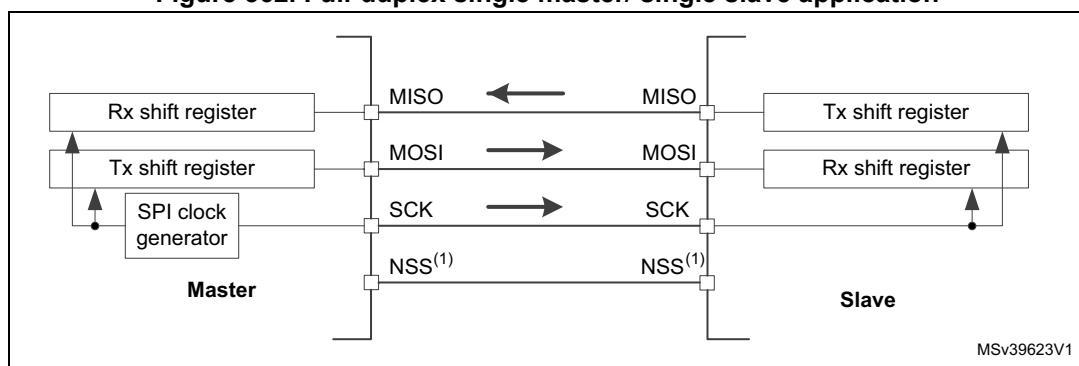
### 35.4.2 Communications between one master and one slave

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software NSS management) or 3 or 4 wires (with hardware NSS management). Communication is always initiated by the master.

#### Full-duplex communication

By default, the SPI is configured for full-duplex communication. In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During SPI communication, data is shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

**Figure 362. Full-duplex single master/ single slave application**

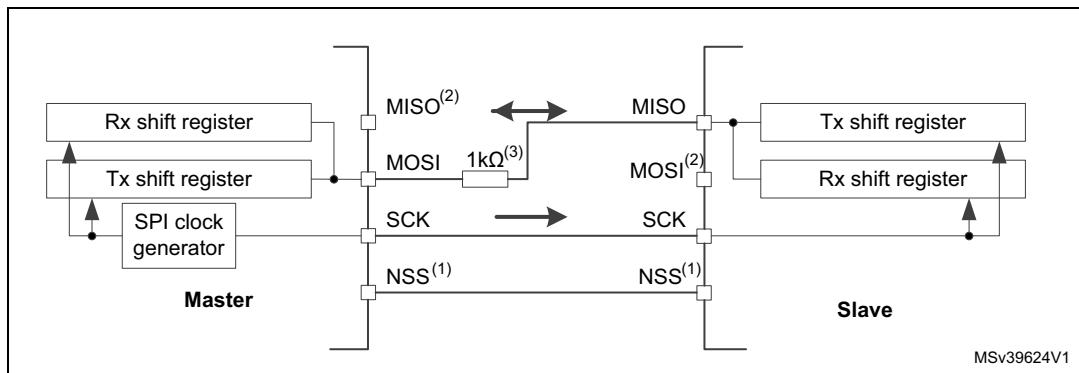


1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 35.4.5: Slave select \(NSS\) pin management](#).

## Half-duplex communication

The SPI can communicate in half-duplex mode by setting the BIDIMODE bit in the SPIx\_CR1 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the BDIOE bit in their SPIx\_CR1 registers. In this configuration, the master's MISO pin and the slave's MOSI pin are free for other application uses and act as GPIOs.

Figure 363. Half-duplex single master/ single slave application



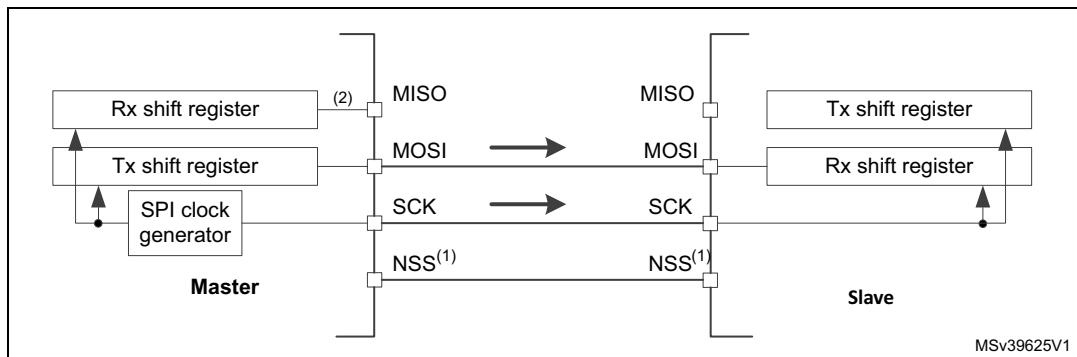
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 35.4.5: Slave select \(NSS\) pin management](#).
2. In this configuration, the master's MISO pin and the slave's MOSI pin can be used as GPIOs.
3. A critical situation can happen when communication direction is changed not synchronously between two nodes working at bidirectional mode and new transmitter accesses the common data line while former transmitter still keeps an opposite value on the line (the value depends on SPI configuration and communication data). Both nodes then fight while providing opposite output levels on the common line temporary till next node changes its direction settings correspondingly, too. It is suggested to insert a serial resistance between MISO and MOSI pins at this mode to protect the outputs and limit the current flowing between them at this situation.

## Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx\_CR1 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0):** The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.
- **Receive-only mode (RXONLY=1):** The application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see [35.4.5: Slave select \(NSS\) pin management](#)). Received data events appear depending on the data buffer configuration. In the master configuration, the MOSI output is disabled and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled. The only way to stop the clock is to clear the RXONLY bit or the SPE bit and wait until the incoming pattern from the MISO pin is finished and fills the data buffer structure, depending on its configuration.

**Figure 364. Simplex single master/single slave application (master in transmit-only/slave in receive-only mode)**



1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 35.4.5: Slave select \(NSS\) pin management](#).
2. An accidental input information is captured at the input of transmitter Rx shift register. All the events associated with the transmitter receive flow must be ignored in standard transmit only mode (e.g. OVR flag).
3. In this configuration, both the MISO pins can be used as GPIOs.

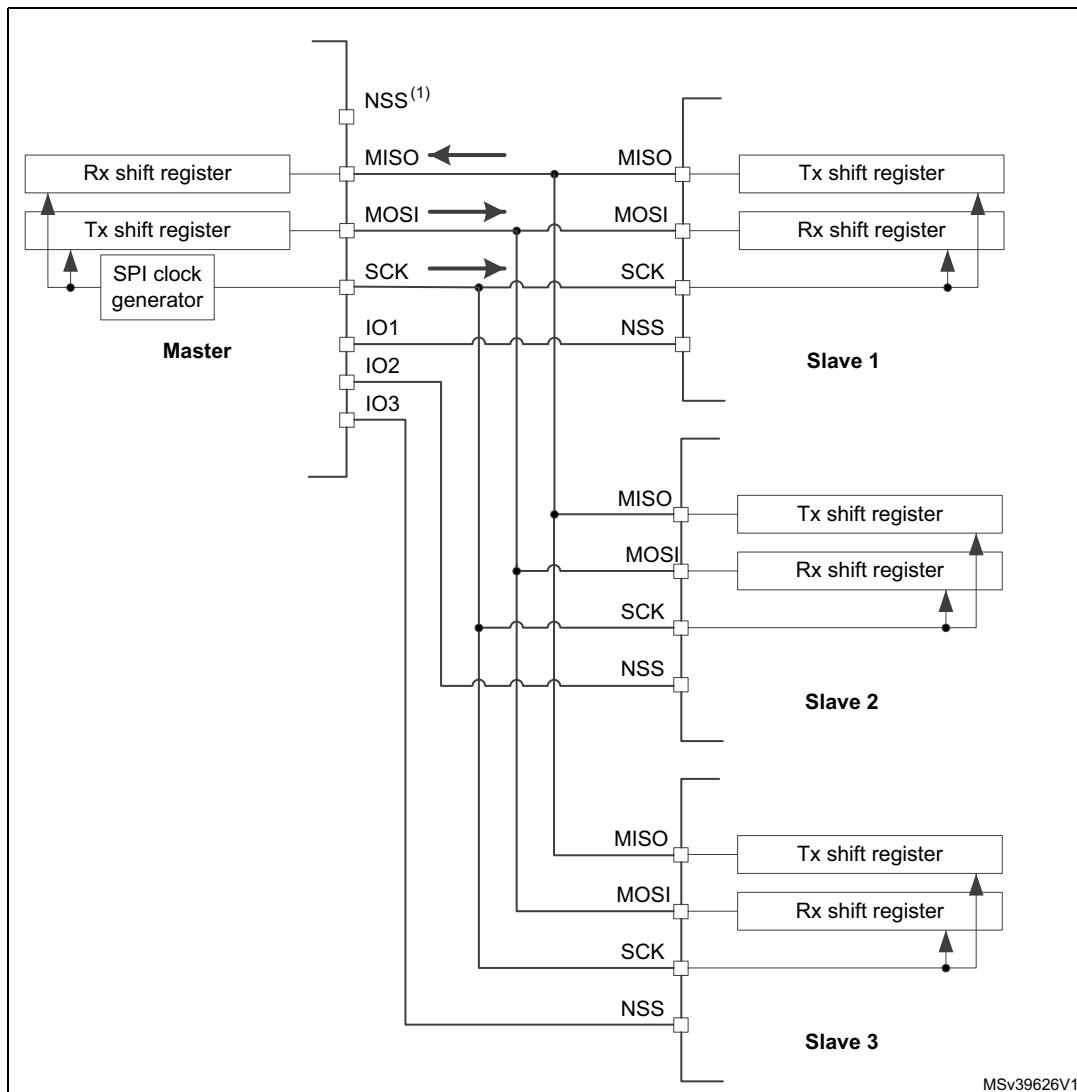
*Note:*

*Any simplex communication can be alternatively replaced by a variant of the half-duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).*

### 35.4.3 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses GPIO pins to manage the chip select lines for each slave (see [Figure 365](#)). The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input. When this is done, a standard master and dedicated slave communication is established.

Figure 365. Master and three independent slaves



1. NSS pin is not used on master side at this configuration. It has to be managed internally (SSM=1, SSI=1) to prevent any MODF error.
2. As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see I/O alternate function input/output section (GPIO)).

### 35.4.4 Multi-master communication

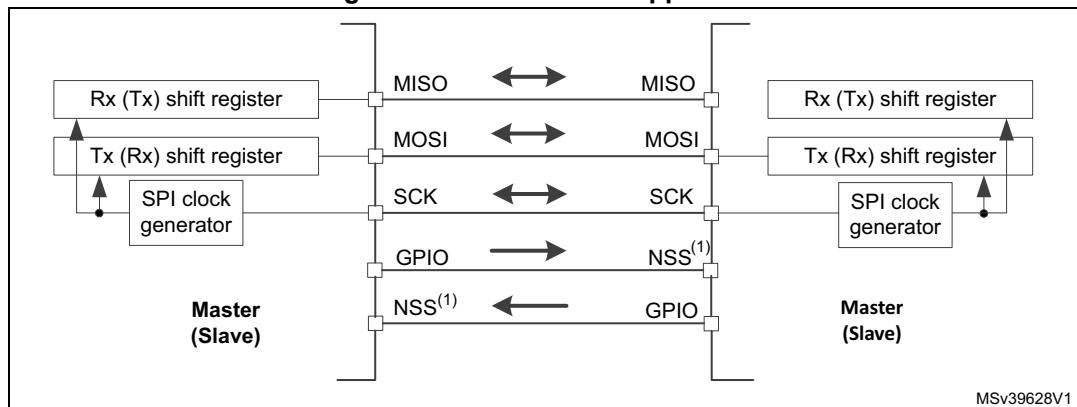
Unless SPI bus is not designed for a multi-master capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used configured at hardware input mode.

The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

When nodes are non active, both stay at slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for next session start.

If potentially both nodes raised their mastering request at the same time a bus conflict event appears (see mode fault MODF event). Then the user can apply some simple arbitration process (e.g. to postpone next attempt by predefined different time-outs applied at both nodes).

Figure 366. Multi-master application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.

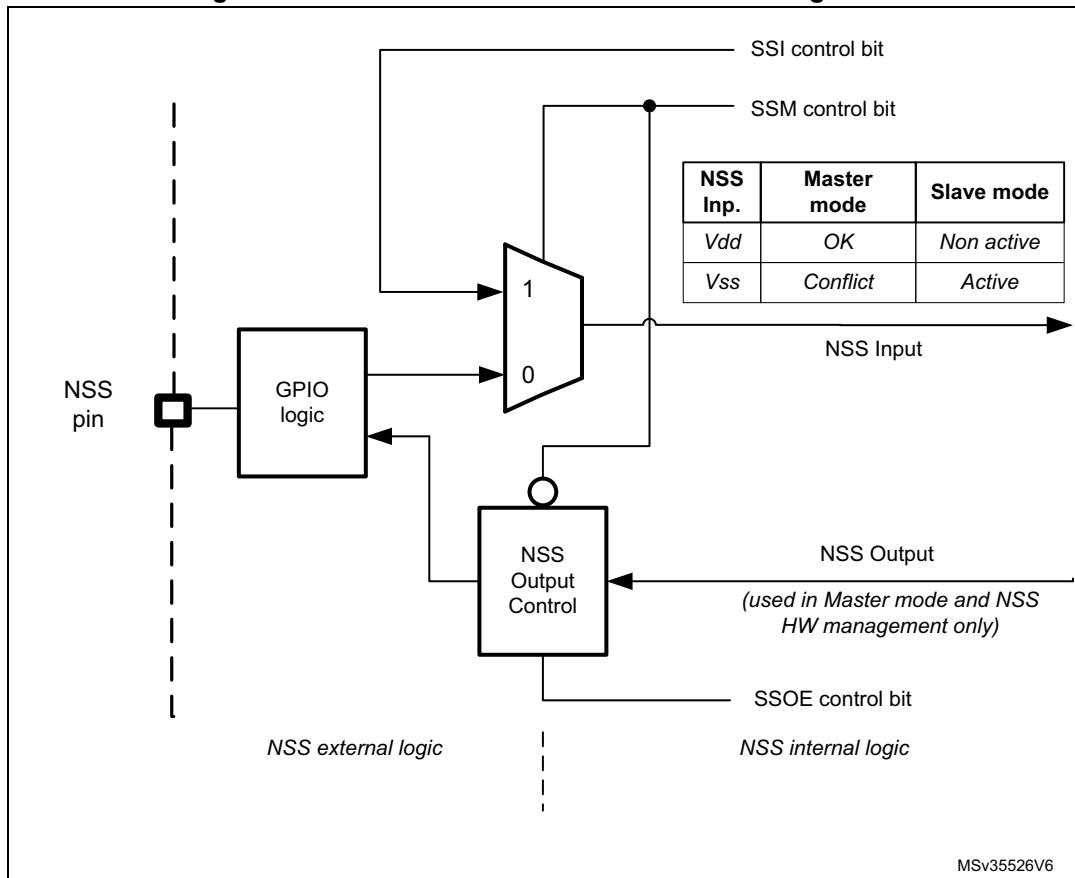
### 35.4.5 Slave select (NSS) pin management

In slave mode, the NSS works as a standard “chip select” input and lets the slave communicate with the master. In master mode, NSS can be used either as output or input. As an input it can prevent multimaster bus collision, and as an output it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SSM bit in the SPIx\_CR1 register:

- **Software NSS management (SSM = 1):** in this configuration, slave select information is driven internally by the SSI bit value in register SPIx\_CR1. The external NSS pin is free for other application uses.
- **Hardware NSS management (SSM = 0):** in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in register SPIx\_CR1).
  - **NSS output enable (SSM=0,SSOE = 1):** this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (SPE=1), and is kept low until the SPI is disabled (SPE =0). A pulse can be generated between continuous communications if NSS pulse mode is activated (NSSP=1). The SPI cannot work in multimaster configuration with this NSS setting.
  - **NSS output disable (SSM=0, SSOE = 0):** if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters master mode fault state and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while NSS line is at low level.

Figure 367. Hardware/software slave select management



### 35.4.6 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

#### Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx\_CR1 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

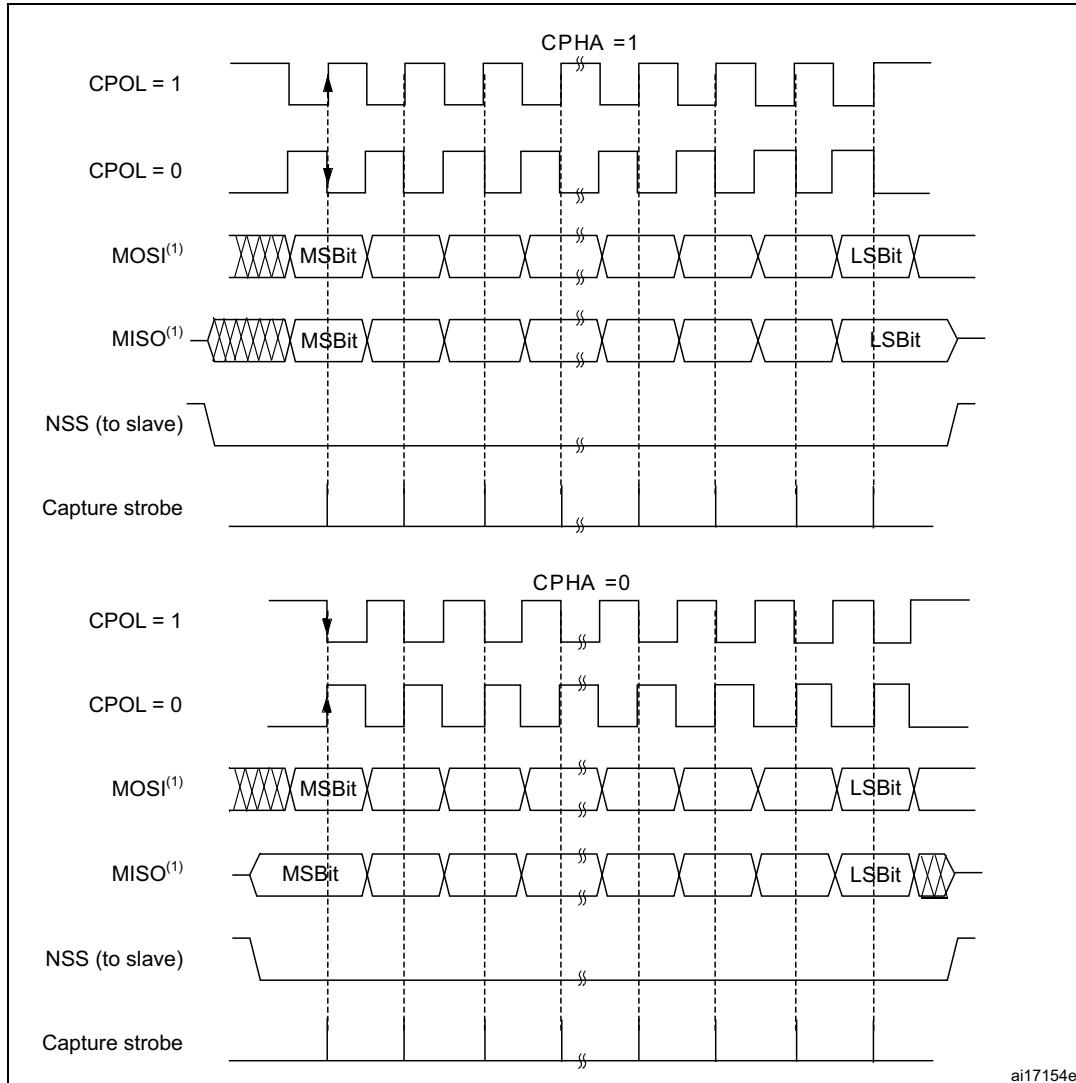
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

**Figure 368**, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

**Note:** *Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.*

*The idle state of SCK must correspond to the polarity selected in the SPIx\_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).*

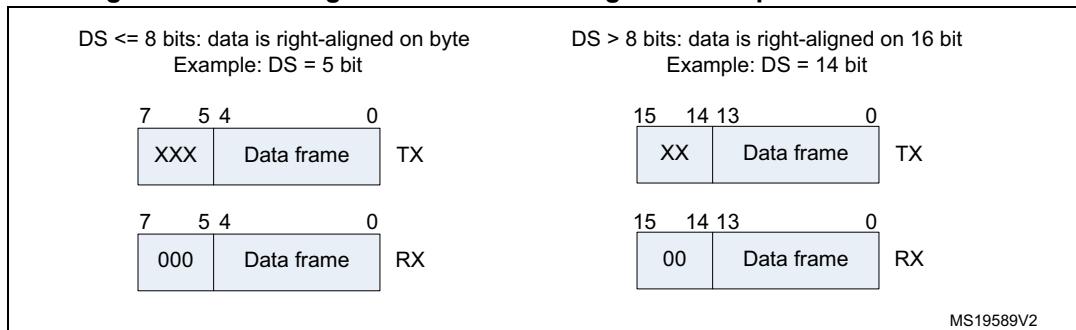
**Figure 368. Data clock timing diagram**



1. The order of data bits depends on LSBFIRST bit setting.

### Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFIRST bit. The data frame size is chosen by using the DS bits. It can be set from 4-bit up to 16-bit length and the setting applies for both transmission and reception. Whatever the selected data frame size, read access to the FIFO must be aligned with the FRXTH level. When the SPIx\_DR register is accessed, data frames are always right-aligned into either a byte (if the data fits into a byte) or a half-word (see [Figure 369](#)). During communication, only bits within the data frame are clocked and transferred.

**Figure 369. Data alignment when data length is not equal to 8-bit or 16-bit**

**Note:** The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 8-bit data frame size.

### 35.4.7 Configuration of SPI

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated sections. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI\_CR1 register:
  - a) Configure the serial clock baud rate using the BR[2:0] bits (Note: 4).
  - b) Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock (CPHA must be cleared in NSSP mode). (Note: 2 - except the case when CRC is enabled at TI mode).
  - c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE cannot be set at the same time).
  - d) Configure the LSBFIRST bit to define the frame format (Note: 2).
  - e) Configure the CRCL and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
  - f) Configure SSM and SSI (Notes: 2 & 3).
  - g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
3. Write to SPI\_CR2 register:
  - a) Configure the DS[3:0] bits to select the data length for the transfer.
  - b) Configure SSOE (Notes: 1 & 2 & 3).
  - c) Set the FRF bit if the TI protocol is required (keep NSSP bit cleared in TI mode).
  - d) Set the NSSP bit if the NSS pulse mode between two data units is required (keep CHPA and TI bits cleared in NSSP mode).
  - e) Configure the RXTH bit. The RXFIFO threshold must be aligned to the read access size for the SPIx\_DR register.
  - f) Initialize LDMA\_TX and LDMA\_RX bits if DMA is used in packed mode.
4. Write to SPI\_CRCPR register: Configure the CRC polynomial if needed.
5. Write proper DMA registers: Configure DMA streams dedicated for SPI Tx and Rx in DMA registers if the DMA streams are used.

- Note:**
- (1) Step is not required in slave mode.
  - (2) Step is not required in TI mode.
  - (3) Step is not required in NSSP mode.
  - (4) The step is not required in slave mode except slave working at TI mode

### 35.4.8 Procedure for enabling SPI

It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

The master at full-duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled and TXFIFO is not empty, or with the next write to TXFIFO.

In any master receive only mode (RXONLY=1 or BIDIMODE=1 & BIDIOE=0), master starts to communicate and the clock starts running immediately after SPI is enabled.

For handling DMA, follow the dedicated section.

### 35.4.9 Data transmission and reception procedures

#### RXFIFO and TXFIFO

All SPI data transactions pass through the 32-bit embedded FIFOs. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each direction has its own FIFO called TXFIFO and RXFIFO. These FIFOs are used in all SPI modes except for receiver-only mode (slave or master) with CRC calculation enabled (see [Section 35.4.14: CRC calculation](#)).

The handling of FIFOs depends on the data exchange mode (duplex, simplex), data frame format (number of bits in the frame), access size performed on the FIFO data registers (8-bit or 16-bit), and whether or not data packing is used when accessing the FIFOs (see [Section 35.4.13: TI mode](#)).

A read access to the SPIx\_DR register returns the oldest value stored in RXFIFO that has not been read yet. A write access to the SPIx\_DR stores the written data in the TXFIFO at the end of a send queue. The read access must be always aligned with the RXFIFO threshold configured by the FRXTH bit in SPIx\_CR2 register. FTLVL[1:0] and FRLVL[1:0] bits indicate the current occupancy level of both FIFOs.

A read access to the SPIx\_DR register must be managed by the RXNE event. This event is triggered when data is stored in RXFIFO and the threshold (defined by FRXTH bit) is reached. When RXNE is cleared, RXFIFO is considered to be empty. In a similar way, write access of a data frame to be transmitted is managed by the TXE event. This event is triggered when the TXFIFO level is less than or equal to half of its capacity. Otherwise TXE is cleared and the TXFIFO is considered as full. In this way, RXFIFO can store up to four data frames, whereas TXFIFO can only store up to three when the data frame format is not greater than 8 bits. This difference prevents possible corruption of 3x 8-bit data frames already stored in the TXFIFO when software tries to write more data in 16-bit mode into TXFIFO. Both TXE and RXNE events can be polled or handled by interrupts. See [Figure 371](#) through [Figure 374](#).

Another way to manage the data exchange is to use DMA (see [Communication using DMA \(direct memory addressing\)](#)).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at [Section 35.4.10: SPI status flags](#)). An overrun event can be polled or handled by an interrupt.

The BSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

### Sequence handling

A few data frames can be passed at single sequence to complete a message. When transmission is enabled, a sequence begins and continues while any data is present in the TXFIFO of the master. The clock signal is provided continuously by the master until TXFIFO becomes empty, then it stops waiting for additional data.

In receive-only modes, half-duplex (BIDIMODE=1, BIDIOE=0) or simplex (BIDIMODE=0, RXONLY=1) the master starts the sequence immediately when both SPI is enabled and receive-only mode is activated. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no underflow error signal for master or slave in SPI mode, and data from the slave is always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislide system to select just one of the slaves for communication. In a single slave system it is not necessary to control the slave with NSS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. NSS can be managed by both software and hardware (see [Section 35.4.5: Slave select \(NSS\) pin management](#)).

When the BSY bit is set it signifies an ongoing data frame transaction. When the dedicated frame transaction is finished, the RXNE flag is raised. The last bit is just sampled and the complete data frame is stored in the RXFIFO.

### Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Ongoing transactions can be corrupted in this case. In some modes the disable procedure is the only way to stop continuous communication running.

Master in full-duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. Special care must be taken in packing mode when an odd number of data frames are transacted to prevent some dummy byte exchange (refer to [Data packing](#) section). Before the SPI is disabled in these modes, the user must follow standard disable procedure. When

the SPI is disabled at the master transmitter while a frame transaction is ongoing or next data frame is stored in TXFIFO, the SPI behavior is not guaranteed.

When the master is in any receive only mode, the only way to stop the continuous clock is to disable the peripheral by SPE=0. This must occur in specific time window within last data frame transaction just between the sampling time of its first bit and before its last bit transfer starts (in order to receive a complete number of expected data frames and to prevent any additional “dummy” data reading after the last valid data frame). Specific procedure must be followed when disabling SPI in this mode.

Data received but not read remains stored in RXFIFO when the SPI is disabled, and must be processed the next time the SPI is enabled, before starting a new sequence. To prevent having unread data, ensure that RXFIFO is empty when disabling the SPI, by using the correct disabling procedure, or by initializing all the SPI registers with a software reset via the control of a specific register dedicated to peripheral reset (see the SPIiRST bits in the RCC\_APBiRSTR registers).

Standard disable procedure is based on pulling BSY status together with FTLVL[1:0] to check if a transmission session is fully completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When NSS signal is managed by software and master has to provide proper end of NSS pulse for slave, or
- When transactions’ streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure is (except when receive only mode is used):

1. Wait until FTLVL[1:0] = 00 (no more data to transmit).
2. Wait until BSY=0 (the last data frame is processed).
3. Disable the SPI (SPE=0).
4. Read data until FRLVL[1:0] = 00 (read all the received data).

The correct disable procedure for certain receive only modes is:

1. Interrupt the receive flow by disabling SPI (SPE=0) in the specific time window while the last data frame is ongoing.
2. Wait until BSY=0 (the last data frame is processed).
3. Read data until FRLVL[1:0] = 00 (read all the received data).

*Note:*

*If packing mode is used and an odd number of data frames with a format less than or equal to 8 bits (fitting into one byte) has to be received, FRXTH must be set when FRLVL[1:0] = 01, in order to generate the RXNE event to read the last odd data frame and to keep good FIFO pointer alignment.*

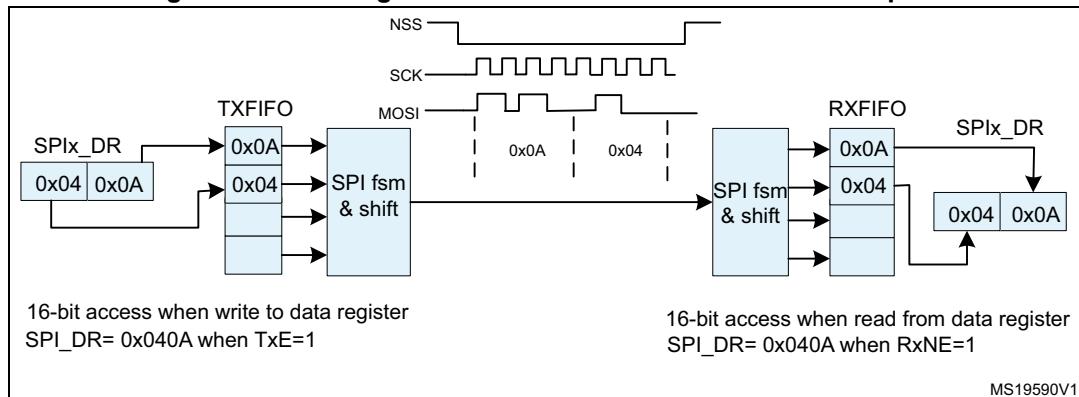
## Data packing

When the data frame size fits into one byte (less than or equal to 8 bits), data packing is used automatically when any read or write 16-bit access is performed on the SPIx\_DR register. The double data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other half stored in the MSB. [Figure 370](#) provides an example of data packing mode sequence handling. Two data frames are sent after the single 16-bit access the SPIx\_DR register of the transmitter. This sequence can generate just one RXNE event in the receiver if the RXFIFO threshold is set to 16 bits (FRXTH=0). The receiver then has to access both data frames by a single 16-bit read of SPIx\_DR as a response to this single RXNE event. The

RxFIFO threshold setting and the following read access must be always kept aligned at the receiver side, as data can be lost if it is not in line.

A specific problem appears if an odd number of such “fit into one byte” data frames must be handled. On the transmitter side, writing the last data frame of any odd sequence with an 8-bit access to SPIx\_DR is enough. The receiver has to change the Rx\_FIFO threshold level for the last data frame received in the odd sequence of frames in order to generate the RXNE event.

**Figure 370. Packing data in FIFO for transmission and reception**



### Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bit in the SPIx\_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx\_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx\_DR register.

See [Figure 371](#) through [Figure 374](#).

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received is not read. When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA\_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. The software must first wait until FTLVL[1:0]=00 and then until BSY=0.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI\_CR2 register, if DMA Rx is used.
2. Enable DMA streams for Tx and Rx in DMA registers, if the streams are used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI\_CR2 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:

1. Disable DMA streams for Tx and Rx in the DMA registers, if the streams are used.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI\_CR2 register, if DMA Tx and/or DMA Rx are used.

### Packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPIx\_CR2 register) packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel. If the DMA channel PSIZE value is equal to 16-bit and SPI data size is less than or equal to 8-bit, then packing mode is enabled. The DMA then automatically manages the write operations to the SPIx\_DR register.

If data packing mode is used and the number of data to transfer is not a multiple of two, the LDMA\_TX/LDMA\_RX bits must be set. The SPI then considers only one data for the transmission or reception to serve the last DMA transfer (for more details refer to [Data packing on page 1188](#).)

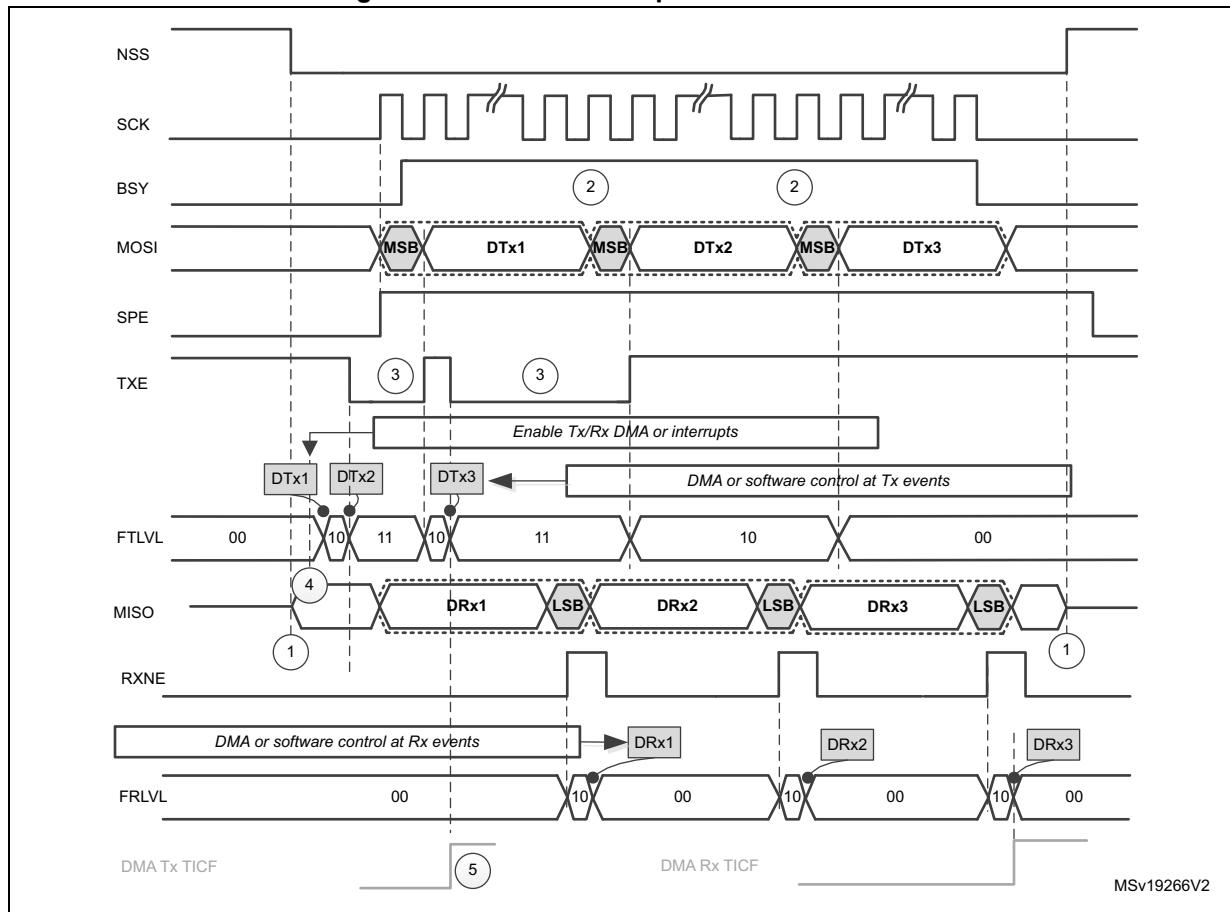
## Communication diagrams

Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by polling, interrupts or DMA. For simplicity, the LSBFIRST=0, CPOL=0 and CPHA=1 setting is used as a common assumption here. No complete configuration of DMA streams is provided.

The following numbered notes are common for [Figure 371 on page 1192](#) through [Figure 374 on page 1195](#):

1. The slave starts to control MISO line as NSS is active and SPI is enabled, and is disconnected from the line when one of them is released. Sufficient time must be provided for the slave to prepare data dedicated to the master in advance before its transaction starts.  
At the master, the SPI peripheral takes control at MOSI and SCK signals (occasionally at NSS signal as well) only if SPI is enabled. If SPI is disabled the SPI peripheral is disconnected from GPIO logic, so the levels at these lines depends on GPIO setting exclusively.
2. At the master, BSY stays active between frames if the communication (clock signal) is continuous. At the slave, BSY signal always goes down for at least one clock cycle between data frames.
3. The TXE signal is cleared only if TXFIFO is full.
4. The DMA arbitration process starts just after the TXDMAEN bit is set. The TXE interrupt is generated just after the TXEIE is set. As the TXE signal is at an active level, data transfers to TxFIFO start, until TxFIFO becomes full or the DMA transfer completes.
5. If all the data to be sent can fit into TxFIFO, the DMA Tx TCIF flag can be raised even before communication on the SPI bus starts. This flag always rises before the SPI transaction is completed.
6. The CRC value for a package is calculated continuously frame by frame in the SPIx\_TXCRCR and SPIx\_RXCRCR registers. The CRC information is processed after the entire data package has completed, either automatically by DMA (Tx channel must be set to the number of data frames to be processed) or by SW (the user must handle CRCNEXT bit during the last data frame processing).  
While the CRC value calculated in SPIx\_TXCRCR is simply sent out by transmitter, received CRC information is loaded into RxFIFO and then compared with the SPIx\_RXCRCR register content (CRC error flag can be raised here if any difference). This is why the user must take care to flush this information from the FIFO, either by software reading out all the stored content of RxFIFO, or by DMA when the proper number of data frames is preset for Rx channel (number of data frames + number of CRC frames) (see the settings at the example assumption).
7. In data packed mode, TxE and RxNE events are paired and each read/write access to the FIFO is 16 bits wide until the number of data frames are even. If the TxFIFO is  $\frac{3}{4}$  full FTLVL status stays at FIFO full level. That is why the last odd data frame cannot be stored before the TxFIFO becomes  $\frac{1}{2}$  full. This frame is stored into TxFIFO with an 8-bit access either by software or automatically by DMA when LDMA\_TX control is set.
8. To receive the last odd data frame in packed mode, the Rx threshold must be changed to 8-bit when the last data frame is processed, either by software setting FRXTH=1 or automatically by a DMA internal signal when LDMA\_RX is set.

Figure 371. Master full-duplex communication



Assumptions for master full-duplex communication example:

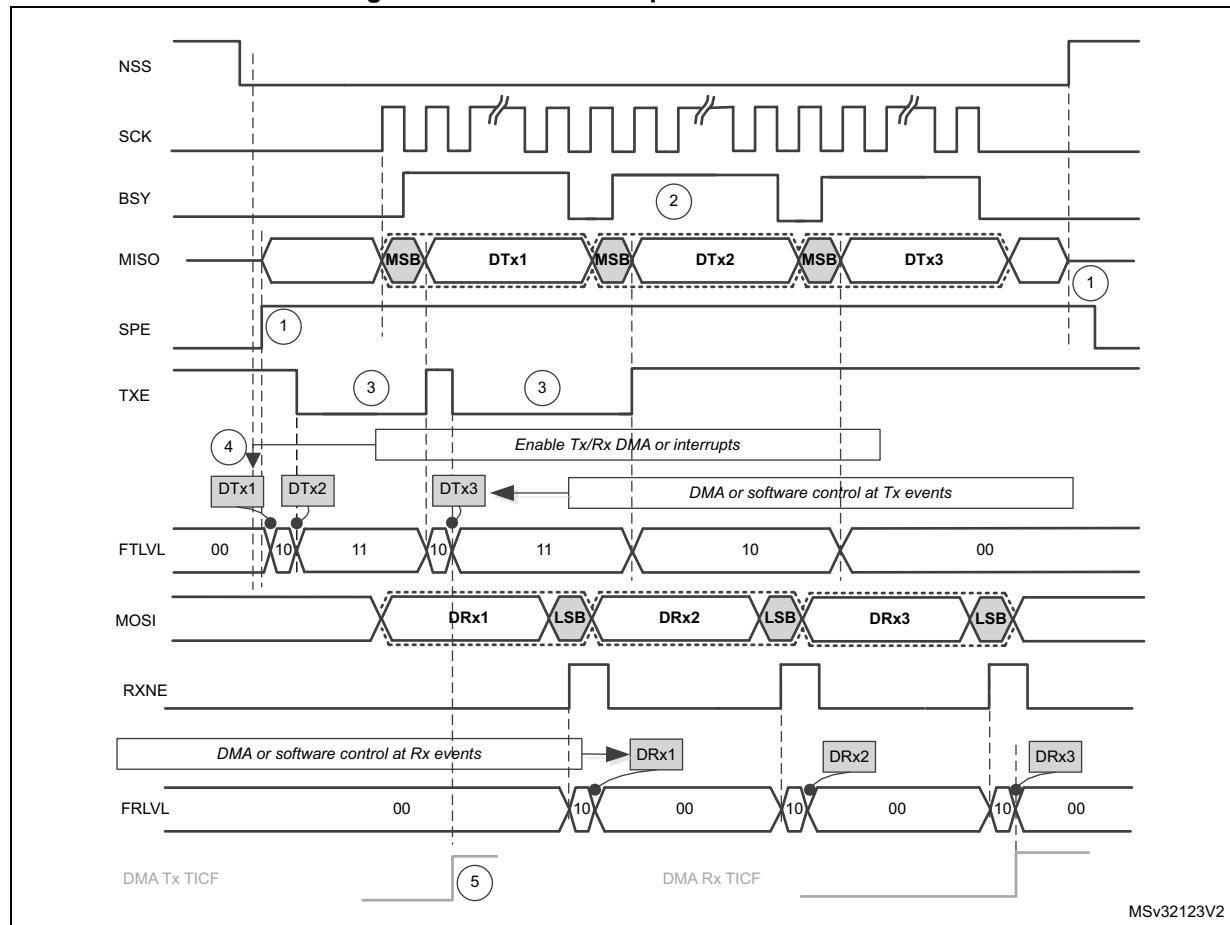
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also : [Communication diagrams on page 1191](#) for details about common assumptions and notes.

Figure 372. Slave full-duplex communication



Assumptions for slave full-duplex communication example:

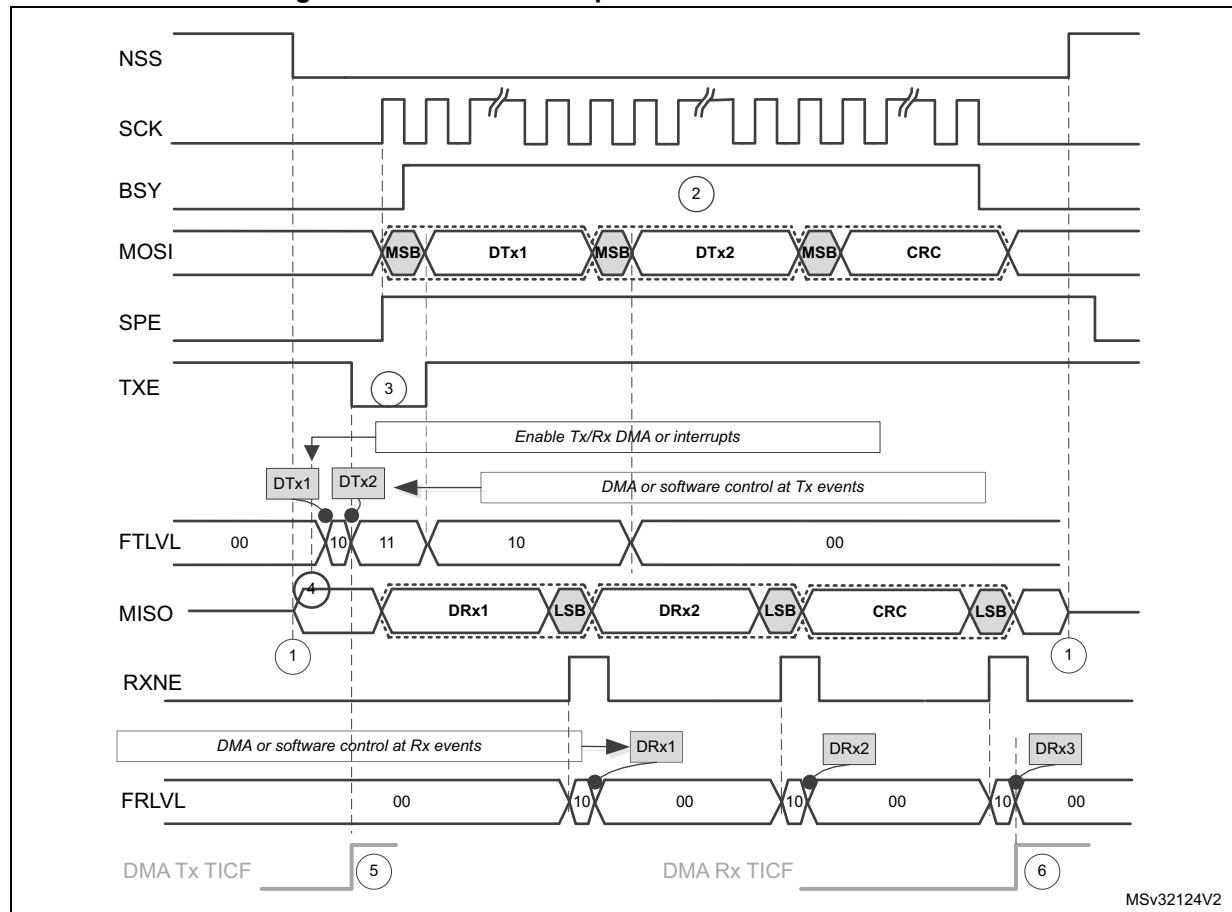
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also [Communication diagrams on page 1191](#) for details about common assumptions and notes.

Figure 373. Master full-duplex communication with CRC



Assumptions for master full-duplex communication with CRC example:

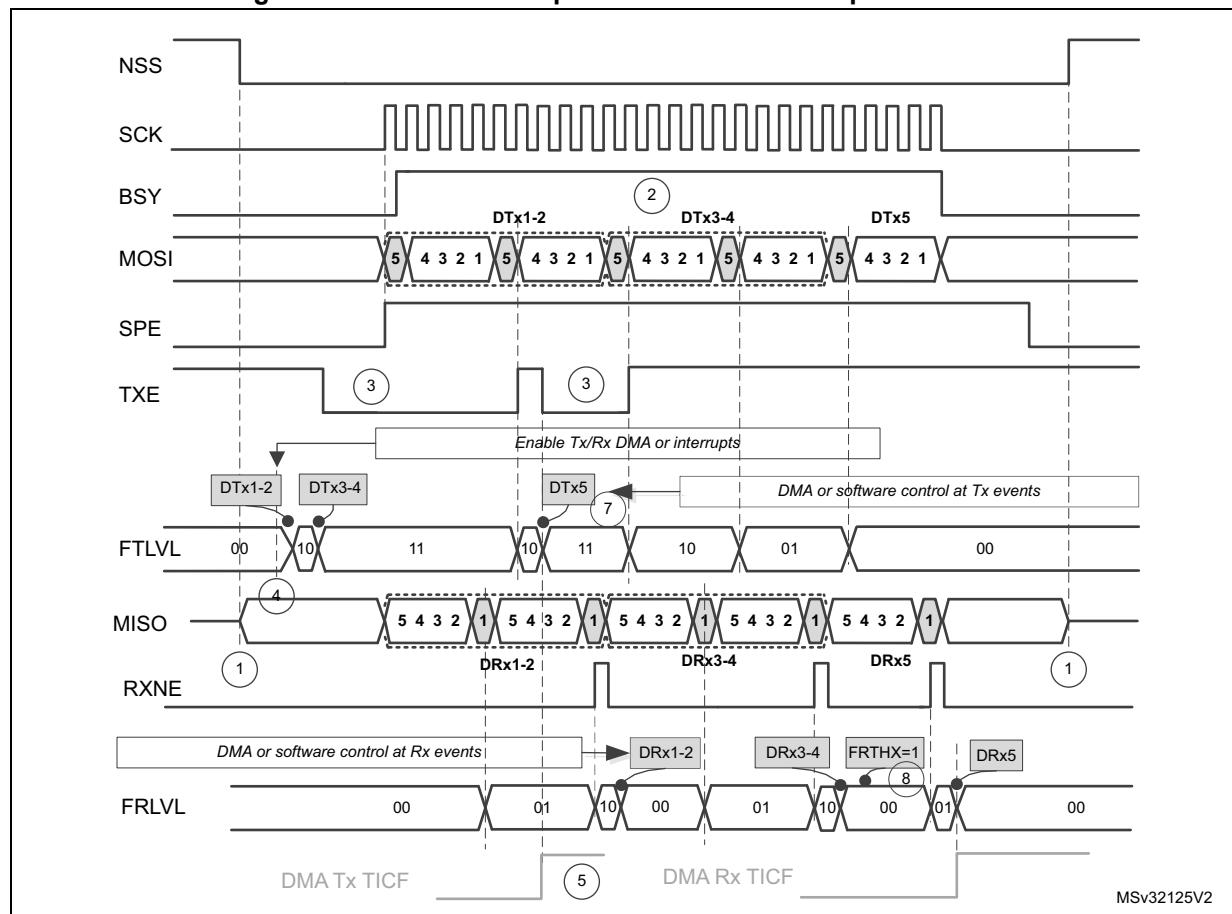
- Data size = 16 bit
- CRC enabled

If DMA is used:

- Number of Tx frames transacted by DMA is set to 2
- Number of Rx frames transacted by DMA is set to 3

See also : [Communication diagrams on page 1191](#) for details about common assumptions and notes.

Figure 374. Master full-duplex communication in packed mode



Assumptions for master full-duplex communication in packed mode example:

- Data size = 5 bit
- Read/write FIFO is performed mostly by 16-bit access
- FRXTH=0

If DMA is used:

- Number of Tx frames to be transacted by DMA is set to 3
- Number of Rx frames to be transacted by DMA is set to 3
- PSIZE for both Tx and Rx DMA channel is set to 16-bit
- LDMA\_TX=1 and LDMA\_RX=1

See also : [Communication diagrams on page 1191](#) for details about common assumptions and notes.

### 35.4.10 SPI status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

#### Tx buffer empty flag (TXE)

The TXE flag is set when transmission TXFIFO has enough space to store data to send. TXE flag is linked to the TXFIFO level. The flag goes high and stays high until the TXFIFO level is lower or equal to 1/2 of the FIFO depth. An interrupt can be generated if the TXEIE bit in the SPIx\_CR2 register is set. The bit is cleared automatically when the TXFIFO level becomes greater than 1/2.

#### Rx buffer not empty (RXNE)

The RXNE flag is set depending on the FRXTH bit value in the SPIx\_CR2 register:

- If FRXTH is set, RXNE goes high and stays high until the RXFIFO level is greater or equal to 1/4 (8-bit).
- If FRXTH is cleared, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/2 (16-bit).

An interrupt can be generated if the RXNEIE bit in the SPIx\_CR2 register is set.

The RXNE is cleared by hardware automatically when the above conditions are no longer true.

#### Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect).

When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).

The BSY flag can be used in certain modes to detect the end of a transfer so that the software can disable the SPI or its peripheral clock before entering a low-power mode which does not provide a clock for the peripheral. This avoids corrupting the last transfer.

The BSY flag is also useful for preventing write collisions in a multimaster system.

The BSY flag is cleared under any one of the following conditions:

- When the SPI is correctly disabled
- When a fault is detected in Master mode (MODF bit set to 1)
- In Master mode, when it finishes a data transmission and no new data is ready to be sent
- In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

Note:

*When the next transmission can be handled immediately by the master (e.g. if the master is in Receive-only mode or its Transmit FIFO is not empty), communication is continuous and the BSY flag remains set to '1' between transfers on the master side. Although this is not the case with a slave, it is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.*

### 35.4.11 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERRIE bit.

#### Overrun flag (OVR)

An overrun condition occurs when data is received by a master or slave and the RXFIFO has not enough space to store this received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RXFIFO) or when space for data storage is limited e.g. the RXFIFO is not available when CRC is enabled in receive only mode so in this case the reception buffer is limited into a single data frame buffer (see [Section 35.4.14: CRC calculation](#)).

When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO. The newly received value is discarded and all data transmitted subsequently is lost. Clearing the OVR bit is done by a read access to the SPI\_DR register followed by a read access to the SPI\_SR register.

#### Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit. Master mode fault affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPIx\_SR register while the MODF bit is set.
2. Then write to the SPIx\_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence. As a security, hardware does not allow the SPE and MSTR bits to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multimaster conflict.

#### CRC error (CRCERR)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx\_CR1 register is set. The CRCERR flag in the SPIx\_SR register is set if the value received in the shift register does not match the receiver SPIx\_RXCRCR value. The flag is cleared by the software.

#### TI mode frame format error (FRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPIx\_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of two data bytes.

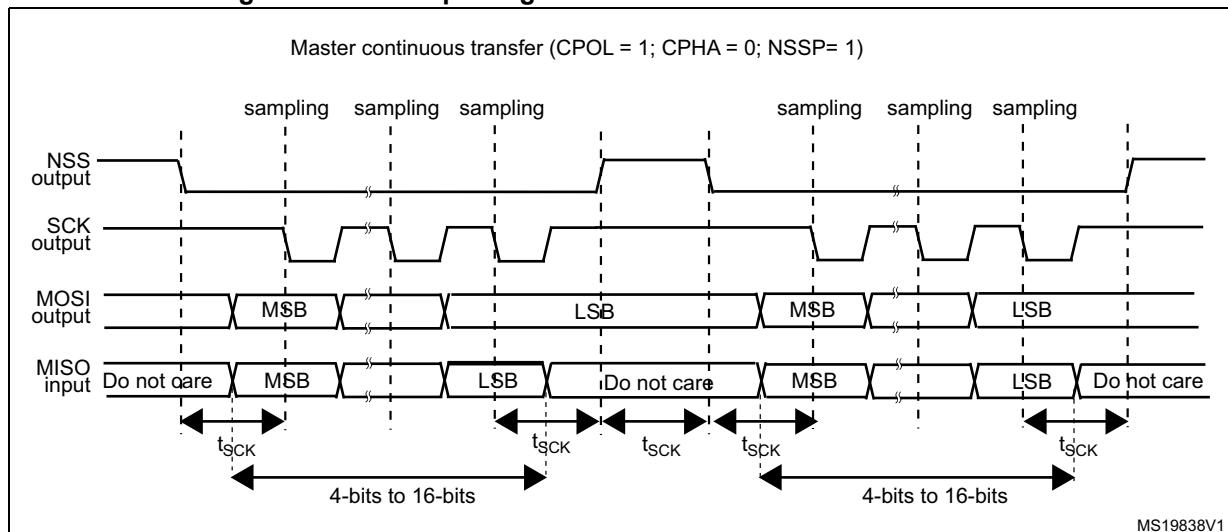
The FRE flag is cleared when SPIx\_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

### 35.4.12 NSS pulse mode

This mode is activated by the NSSP bit in the SPIx\_CR2 register and it takes effect only if the SPI interface is configured as Motorola SPI master (FRF=0) with capture on the first edge (SPIx\_CR1 CPHA = 0, CPOL setting is ignored). When activated, an NSS pulse is generated between two consecutive data frame transfers when NSS stays at high level for the duration of one clock period at least. This mode allows the slave to latch data. NSSP pulse mode is designed for applications with a single master-slave pair.

*Figure 375* illustrates NSS pin management when NSSP pulse mode is enabled.

**Figure 375. NSSP pulse generation in Motorola SPI master mode**



**Note:** Similar behavior is encountered when CPOL = 0. In this case the sampling edge is the *rising* edge of SCK, and NSS assertion and deassertion refer to this sampling edge.

### 35.4.13 TI mode

#### TI protocol in master mode

The SPI interface is compatible with the TI protocol. The FRF bit of the SPIx\_CR2 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPIx\_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPIx\_CR1 and SPIx\_CR2 registers (SSM, SSI, SSOE) impossible in this case.

In slave mode, the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ when the current transaction finishes (see *Figure 376*). Any baud rate can be used, making it possible to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The delay for the MISO signal to become HiZ ( $t_{release}$ ) depends on internal resynchronization and on the

baud rate value set in through the BR[2:0] bits in the SPIx\_CR1 register. It is given by the formula:

$$\frac{t_{baud\_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud\_rate}}{2} + 6 \times t_{pclk}$$

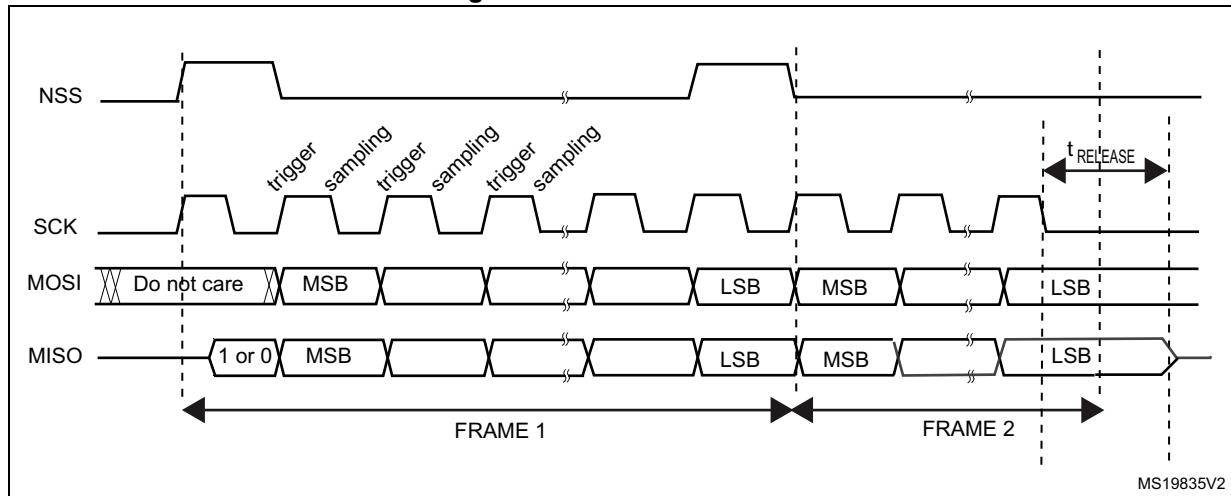
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

If the data size is equal to 4-bits or 5-bits, the master in full-duplex mode or transmit-only mode uses a protocol with one more dummy data bit added after LSB. TI NSS pulse is generated above this dummy bit clock cycle instead of the LSB in each period.

This feature is not available for Motorola SPI communications (FRF bit set to 0).

*Figure 376: TI mode transfer* shows the SPI communication waveforms when TI mode is selected.

Figure 376. TI mode transfer



### 35.4.14 CRC calculation

Two separate CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation independently of the frame data length, which can be fixed to 8-bit or 16-bit. For all the other data frame lengths, no CRC is available.

#### CRC principle

CRC calculation is enabled by setting the CRCEN bit in the SPIx\_CR1 register before the SPI is enabled (SPE = 1). The CRC value is calculated using an odd programmable polynomial on each bit. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx\_CR1 register. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management.

**Note:** *The polynomial value should only be odd. No even values are supported.*

### CRC transfer managed by CPU

Communication starts and continues normally until the last data frame has to be sent or received in the SPIx\_DR register. Then CRCNEXT bit has to be set in the SPIx\_CR1 register to indicate that the CRC frame transaction follows after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the RXFIFO like a data byte or word. That is why in CRC mode only, the reception buffer has to be considered as a single 16-bit buffer used to receive only one data frame at a time.

A CRC-format transaction usually takes one more data frame to communicate at the end of data sequence. However, when setting an 8-bit data frame checked by 16-bit CRC, two more frames are necessary to send the complete CRC.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx\_RXCRC register. Software has to check the CRCERR flag in the SPIx\_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the RXFIFO and must be read in the SPIx\_DR register in order to clear the RXNE flag.

### CRC transfer managed by DMA

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out received CRC information from RXFIFO as it is always loaded into it. In full-duplex mode, the counter of the reception DMA channel can be set to the number of data frames to receive including the CRC, which means, for example, in the specific case of an 8-bit data frame checked by 16-bit CRC:

$$\text{DMA\_RX} = \text{Numb\_of\_data} + 2$$

In receive only mode, the DMA reception channel counter should contain only the amount of data transferred, excluding the CRC calculation. Then based on the complete transfer from DMA, all the CRC values must be read back by software from FIFO as it works as a single buffer in this mode.

At the end of the data and CRC transfers, the CRCERR flag in the SPIx\_SR register is set if corruption occurred during the transfer.

If packing mode is used, the LDMA\_RX bit needs managing if the number of data is odd.

### Resetting the SPIx\_TXCRC and SPIx\_RXCRC values

The SPIx\_TXCRC and SPIx\_RXCRC values are cleared automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode (not available in receive-only mode) in order to transfer data without any interruption, (several data blocks covered by intermediate CRC checking phases).

If the SPI is disabled during a communication the following sequence must be followed:

1. Disable the SPI
2. Clear the CRCEN bit
3. Enable the CRCEN bit
4. Enable the SPI

Note:

*When the SPI interface is configured as a slave, the NSS internal signal needs to be kept low during transaction of the CRC phase once the CRCNEXT signal is released. That is why the CRC calculation cannot be used at NSS Pulse mode when NSS hardware mode should be applied at slave normally.*

*At TI mode, despite the fact that clock phase and clock polarity setting is fixed and independent on SPIx\_CR1 register, the corresponding setting CPOL=0 CPHA=1 has to be kept at the SPIx\_CR1 register anyway if CRC is applied. In addition, the CRC calculation has to be reset between sessions by SPI disable sequence with re-enable the CRCEN bit described above at both master and slave side, else CRC calculation can be corrupted at this specific mode.*

## 35.5 SPI interrupts

During SPI communication an interrupt can be generated by the following events:

- Transmit TXFIFO ready to be loaded
- Data received in Receive RXFIFO
- Master mode fault
- Overrun error
- TI frame format error
- CRC protocol error

Interrupts can be enabled and disabled separately.

**Table 221. SPI interrupt requests**

Interrupt event	Event flag	Enable Control bit
Transmit TXFIFO ready to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
TI frame format error	FRE	
CRC protocol error	CRCERR	

## 35.6 SPI registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI\_DR in addition can be accessed by 8-bit access.

### 35.6.1 SPI control register 1 (SPIx\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDIOE	CRC EN	CRCN EXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDIMODE**: Bidirectional data mode enable.

This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active.

0: 2-line unidirectional data mode selected

1: 1-line bidirectional data mode selected

Bit 14 **BIDIOE**: Output enable in bidirectional mode

This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode.

0: Output disabled (receive-only mode)

1: Output enabled (transmit-only mode)

*Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used.*

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.*

Bit 12 **CRCNEXT**: Transmit CRC next

0: Next transmit value is from Tx buffer.

1: Next transmit value is from Tx CRC register.

*Note: This bit has to be written as soon as the last data is written in the SPIx\_DR register.*

Bit 11 **CRCL**: CRC length

This bit is set and cleared by software to select the CRC length.

0: 8-bit CRC length

1: 16-bit CRC length

*Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation.*

Bit 10 **RXONLY**: Receive only mode enabled.

This bit enables simplex communication using a single unidirectional line to receive data exclusively. Keep BIDIMODE bit clear when receive only mode is active. This bit is also useful in a multislave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.

- 0: Full-duplex (Transmit and receive)
- 1: Output disabled (Receive-only mode)

Bit 9 **SSM**: Software slave management

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

- 0: Software slave management disabled
- 1: Software slave management enabled

*Note: This bit is not used in SPI TI mode.*

Bit 8 **SSI**: Internal slave select

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the I/O value of the NSS pin is ignored.

*Note: This bit is not used in SPI TI mode.*

Bit 7 **LSBFIRST**: Frame format

- 0: data is transmitted / received with the MSB first
- 1: data is transmitted / received with the LSB first

*Note: 1. This bit should not be changed when communication is ongoing.  
2. This bit is not used in SPI TI mode.*

Bit 6 **SPE**: SPI enable

- 0: Peripheral disabled
- 1: Peripheral enabled

*Note: When disabling the SPI, follow the procedure described in [Procedure for disabling the SPI on page 1187](#).*

Bits 5:3 **BR[2:0]**: Baud rate control

- 000:  $f_{PCLK}/2$
- 001:  $f_{PCLK}/4$
- 010:  $f_{PCLK}/8$
- 011:  $f_{PCLK}/16$
- 100:  $f_{PCLK}/32$
- 101:  $f_{PCLK}/64$
- 110:  $f_{PCLK}/128$
- 111:  $f_{PCLK}/256$

*Note: These bits should not be changed when communication is ongoing.*

Bit 2 **MSTR**: Master selection

- 0: Slave configuration
- 1: Master configuration

*Note: This bit should not be changed when communication is ongoing.*

Bit 1 **CPOL:** Clock polarity

- 0: CK to 0 when idle
- 1: CK to 1 when idle

*Note: This bit should not be changed when communication is ongoing.*

*This bit is not used in SPI TI mode except the case when CRC is applied at TI mode.*

Bit 0 **CPHA:** Clock phase

- 0: The first clock transition is the first data capture edge
- 1: The second clock transition is the first data capture edge

*Note: This bit should not be changed when communication is ongoing.*

*This bit is not used in SPI TI mode except the case when CRC is applied at TI mode.*

### 35.6.2 SPI control register 2 (SPIx\_CR2)

Address offset: 0x04

Reset value: 0x0700

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		LDMA_TX	LDMA_RX	FRXTH	DS[3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **LDMA\_TX:** Last DMA transfer for transmission

This bit is used in data packing mode, to define if the total number of data to transmit by DMA is odd or even. It has significance only if the TXDMAEN bit in the SPIx\_CR2 register is set and if packing mode is used (data length <= 8-bit and write access to SPIx\_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx\_CR1 register).

- 0: Number of data to transfer is even
- 1: Number of data to transfer is odd

*Note: Refer to [Procedure for disabling the SPI on page 1187](#) if the CRCEN bit is set.*

Bit 13 **LDMA\_RX:** Last DMA transfer for reception

This bit is used in data packing mode, to define if the total number of data to receive by DMA is odd or even. It has significance only if the RXDMAEN bit in the SPIx\_CR2 register is set and if packing mode is used (data length <= 8-bit and write access to SPIx\_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx\_CR1 register).

- 0: Number of data to transfer is even
- 1: Number of data to transfer is odd

*Note: Refer to [Procedure for disabling the SPI on page 1187](#) if the CRCEN bit is set.*

Bit 12 **FRXTH:** FIFO reception threshold

This bit is used to set the threshold of the RXFIFO that triggers an RXNE event

- 0: RXNE event is generated if the FIFO level is greater than or equal to 1/2 (16-bit)
- 1: RXNE event is generated if the FIFO level is greater than or equal to 1/4 (8-bit)

Bits 11:8 **DS[3:0]**: Data size

These bits configure the data length for SPI transfers.

0000: Not used  
 0001: Not used  
 0010: Not used  
 0011: 4-bit  
 0100: 5-bit  
 0101: 6-bit  
 0110: 7-bit  
 0111: 8-bit  
 1000: 9-bit  
 1001: 10-bit  
 1010: 11-bit  
 1011: 12-bit  
 1100: 13-bit  
 1101: 14-bit  
 1110: 15-bit  
 1111: 16-bit

If software attempts to write one of the “Not used” values, they are forced to the value “0111” (8-bit)

Bit 7 **TXEIE**: Tx buffer empty interrupt enable

0: TXE interrupt masked  
 1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE**: RX buffer not empty interrupt enable

0: RXNE interrupt masked  
 1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE**: Error interrupt enable

This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode).

0: Error interrupt is masked  
 1: Error interrupt is enabled

Bit 4 **FRF**: Frame format

0: SPI Motorola mode  
 1 SPI TI mode

*Note: This bit must be written only when the SPI is disabled (SPE=0).*

Bit 3 **NSSP**: NSS pulse management

This bit is used in master mode only. It allows the SPI to generate an NSS pulse between two consecutive data when doing continuous transfers. In the case of a single data transfer, it forces the NSS pin high level after the transfer.

It has no meaning if CPHA = '1', or FRF = '1'.

0: No NSS pulse  
 1: NSS pulse generated

*Note: 1. This bit must be written only when the SPI is disabled (SPE=0).*

*2. This bit is not used in SPI TI mode.*

Bit 2 **SSOE**: SS output enable

0: SS output is disabled in master mode and the SPI interface can work in multimaster configuration

1: SS output is enabled in master mode and when the SPI interface is enabled. The SPI interface cannot work in a multimaster environment.

*Note: This bit is not used in SPI TI mode.*

Bit 1 **TXDMAEN**: Tx buffer DMA enable

When this bit is set, a DMA request is generated whenever the TXE flag is set.

0: Tx buffer DMA disabled

1: Tx buffer DMA enabled

Bit 0 **RXDMAEN**: Rx buffer DMA enable

When this bit is set, a DMA request is generated whenever the RXNE flag is set.

0: Rx buffer DMA disabled

1: Rx buffer DMA enabled

**35.6.3 SPI status register (SPIx\_SR)**

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		FRE	BSY	OVR	MODF	CRCE RR	Res.	Res.	TXE	RXNE
			r	r	r	r	r	r	r	r	rc_w0			r	r

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:11 **FTLVL[1:0]**: FIFO transmission level

These bits are set and cleared by hardware.

00: FIFO empty

01: 1/4 FIFO

10: 1/2 FIFO

11: FIFO full (considered as FULL when the FIFO threshold is greater than 1/2)

Bits 10:9 **FRLVL[1:0]**: FIFO reception level

These bits are set and cleared by hardware.

00: FIFO empty

01: 1/4 FIFO

10: 1/2 FIFO

11: FIFO full

*Note: These bits are not used in SPI receive-only mode while CRC calculation is enabled.*

Bit 8 **FRE**: Frame format error

This flag is used for SPI in TI slave mode. Refer to [Section 35.4.11: SPI error flags](#).

This flag is set by hardware and reset when SPIx\_SR is read by software.

0: No frame format error

1: A frame format error occurred

Bit 7 **BSY**: Busy flag

0: SPI not busy

1: SPI is busy in communication or Tx buffer is not empty

This flag is set and cleared by hardware.

*Note: The BSY flag must be used with caution: refer to [Section 35.4.10: SPI status flags and Procedure for disabling the SPI](#) on page 1187.*

Bit 6 **OVR**: Overrun flag

0: No overrun occurred

1: Overrun occurred

This flag is set by hardware and reset by a software sequence.

Bit 5 **MODF**: Mode fault

0: No mode fault occurred

1: Mode fault occurred

This flag is set by hardware and reset by a software sequence. Refer to [Section : Mode fault \(MODF\) on page 1197](#) for the software sequence.

Bit 4 **CRCERR**: CRC error flag

0: CRC value received matches the SPIx\_RXCRCR value

1: CRC value received does not match the SPIx\_RXCRCR value

*Note: This flag is set by hardware and cleared by software writing 0.*

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TXE**: Transmit buffer empty

0: Tx buffer not empty

1: Tx buffer empty

Bit 0 **RXNE**: Receive buffer not empty

0: Rx buffer empty

1: Rx buffer not empty

### 35.6.4 SPI data register (SPIx\_DR)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DR[15:0]**: Data register

Data received or to be transmitted

The data register serves as an interface between the Rx and Tx FIFOs. When the data register is read, RxFIFO is accessed while the write to data register accesses TxFIFO (See [Section 35.4.9: Data transmission and reception procedures](#)).

*Note: Data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read. The Rx threshold setting must always correspond with the read access currently used.*

### 35.6.5 SPI CRC polynomial register (SPIx\_CRCPR)

Address offset: 0x10

Reset value: 0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CRCPOLY[15:0]**: CRC polynomial register

This register contains the polynomial for the CRC calculation.

The CRC polynomial (0x0007) is the reset value of this register. Another polynomial can be configured as required.

*Note: The polynomial value should be odd only. No even value is supported.*

### 35.6.6 SPI Rx CRC register (SPIx\_RXCRCR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **RXCRC[15:0]**: Rx CRC register

When CRC calculation is enabled, the RXCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPIx\_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx\_CRCPR register.

Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (CRCL bit in the SPIx\_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx\_CR1 register is set). CRC calculation is done based on any CRC16 standard.

*A read to this register when the BSY Flag is set could return an incorrect value.*

### 35.6.7 SPI Tx CRC register (SPIx\_TXCRCR)

Address offset: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

**Bits 15:0 TXCRC[15:0]: Tx CRC register**

When CRC calculation is enabled, the TXCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPIx\_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx\_CRCPR register.

Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (CRCL bit in the SPIx\_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx\_CR1 register is set). CRC calculation is done based on any CRC16 standard.

*A read to this register when the BSY flag is set could return an incorrect value.*

### 35.6.8 SPI register map

*Table 222* shows the SPI register map and reset values.

**Table 222. SPI register map and reset values**

Offset	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPIx_CR1	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	CRCL	RXONLY	SSM	SSI	LSBFIRST	SPE	ERRIE	FRF	NSSP	MSTR	CPOL	CPHA
	Reset value	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	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	SPIx_CR2	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]	TXEIE	RXNEIE	OVR	MODF	CRCEERR	SSOE	TXDMAEN	1	TXE	0	0	0
	Reset value	0 0 0 0	0 0 0 0	0 0 0 0	0 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
0x08	SPIx_SR	FTLV[1:0]	FTLV[1:0]	FRE	BSY	TXEIE	RXNEIE	ERRIE	CRCEERR	SSOE	TXDMAEN	1	TXE	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 0 0	0 0 0 0	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	SPIx_DR	DR[15:0]															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															
0x10	SPIx_CRCPR	CRCPOLY[15:0]															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															
0x14	SPIx_RXCRCR	RXCRC[15:0]															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															
0x18	SPIx_TXCRCR	TXCRC[15:0]															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 36 Serial audio interface (SAI)

### 36.1 Introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many stereo or mono audio applications may be targeted. I2S standards, LSB or MSB-justified, PCM/DSP, TDM, and AC'97 protocols may be addressed for example. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has its own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

### 36.2 SAI main features

- Two independent audio subblocks which can be transmitters or receivers with their respective FIFO.
- 8-word integrated FIFOs for each audio subblock.
- Synchronous or asynchronous mode between the audio subblocks.
- Master or slave configuration independent for both audio subblocks.
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode.
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit.
- Audio protocol: I2S, LSB or MSB-justified, PCM/DSP, TDM, AC'97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required.
- Up to 16 slots available with configurable size.
- Number of bits by frame can be configurable.
- Frame synchronization active level configurable (offset, bit length, level).
- First active bit position in the slot is configurable.
- LSB first or MSB first for data transfer.
- Mute mode.
- Stereo/Mono audio frame capability.
- Communication clock strobing edge configurable (SCK).
- Error flags with associated interrupts if enabled respectively.
  - Overrun and underrun detection,
  - Anticipated frame synchronization signal detection in slave mode,
  - Late frame synchronization signal detection in slave mode,
  - Codec not ready for the AC'97 mode in reception.
- Interrupt sources when enabled:
  - Errors,
  - FIFO requests.

- 2-channel DMA interface.

## 36.3 SAI implementation

Table 223. STM32WB55xx SAI features <sup>(1)</sup>

SAI features	SAI1
I2S, LSB or MSB-justified, PCM/DSP, TDM, AC'97	X
FIFO size	8 words
SPDIF	X
PDM	X <sup>(2)(3)</sup>

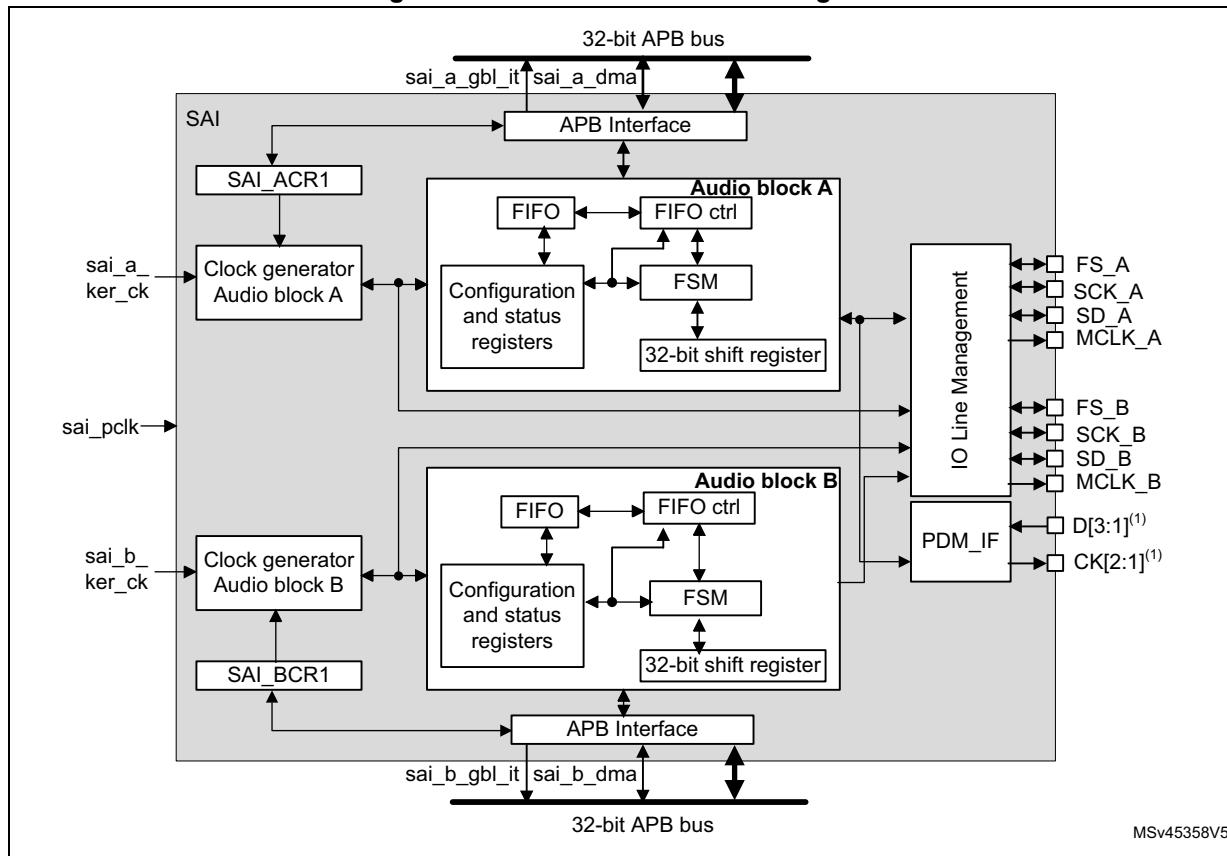
1. 'X' = supported, '-' = not supported.
2. Only signals D[3:1], and CK[2:1] are available.
3. The DLYM4R[2:0] and DLYM4L[2:0] bitfields shall not be used and MICNBR[1:0] shall not be set to 0b11.

## 36.4 SAI functional description

### 36.4.1 SAI block diagram

*Figure 377* shows the SAI block diagram while *Table 224* and *Table 225* list SAI internal and external signals.

Figure 377. SAI functional block diagram



MSv45358V5

1. These signals might not be available for all SAI instances. Refer to [Section 36.3: SAI implementation](#) for details.

The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA in order to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD\_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set-up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK\_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC'97 protocol, it is an SAI output even if the SAI (link controller) is set-up to consume the SCK clock (and so to be in Slave mode).

**Note:** For ease of reading of this section, the notation SAI\_x refers to SAI\_A or SAI\_B, where 'x' represents the SAI A or B subblock.

### 36.4.2 SAI pins and internal signals

**Table 224. SAI internal input/output signals**

Internal signal name	Signal type	Description
sai_a_gbl_it/ sai_b_gbl_it	Output	Audio block A and B global interrupts.
sai_a_dma, sai_b_dma	Input/output	Audio block A and B DMA acknowledges and requests.
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock.
sai_pclk	Input	APB clock.

**Table 225. SAI input/output pins**

Name	Signal type	Comments
SAI_SCK_A/B	Input/output	Audio block A/B bit clock.
SAI_MCLK_A/B	Output	Audio block A/B master clock.
SAI_SD_A/B	Input/output	Data line for block A/B.
SAI_FS_A/B	Input/output	Frame synchronization line for audio block A/B.
SAI_CK[2:1]	Output	PDM bitstream clock <sup>(1)</sup> .
SAI_D[3:1]	Input	PDM bitstream data <sup>(1)</sup> .

1. These signals might not be available in all SAI instances. Refer to [Section 36.3: SAI implementation](#) for details.

### 36.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via MODE bits in the SAI\_xCR1 register of the selected audio block.

#### Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK\_x and FS\_x, respectively.
- If needed, the SAI can also generate a master clock on MCLK\_x pin.

Both SCK\_x, FS\_x and MCLK\_x are configured as outputs.

### Slave mode

The SAI expects to receive timing signals from an external device.

- If the SAI subblock is configured in asynchronous mode, then SCK\_x and FS\_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with the second audio subblock, the corresponding SCK\_x and FS\_x pins are left free to be used as general purpose I/Os.

In slave mode, MCLK\_x pin is not used and can be assigned to another function.

It is recommended to enable the slave device before enabling the master.

### Configuring and enabling SAI modes

Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI\_xCR1 register of the corresponding audio block. As a result, SAI\_SD\_x pin is respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI are enabled by SAIEN bit in the SAI\_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock line, data line and synchronization line in slave mode.

In master TX mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO. However FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave TX mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:

1. Write into the SAI\_xDR (by software or by DMA).
2. Wait until the FIFO threshold (FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

### 36.4.4 SAI synchronization mode

SAI sub-clock A and B can be synchronized.

#### Internal synchronization

An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK\_x, FS\_x, and MCLK\_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS\_x and SCK\_x ad MCLK\_x I/O pins are relevant (if the audio block is considered as master).

Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI\_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI\_xCR1).

*Note:* *Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.*

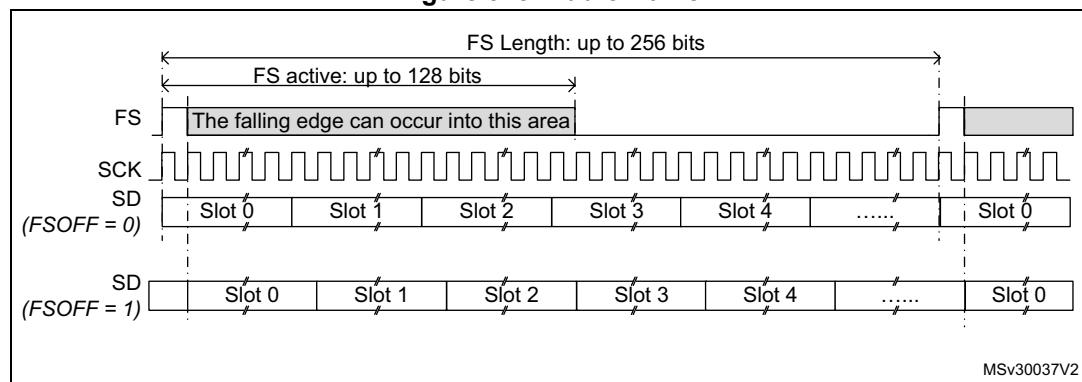
### 36.4.5 Audio data size

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI\_xCR1 register. The data sizes may be 8, 10, 16, 20, 24 or 32 bits. During the transfer, either the MSB or the LSB of the data are sent first, depending on the configuration of bit LSBFIRST in the SAI\_xCR1 register.

### 36.4.6 Frame synchronization

The FS signal acts as the Frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable in order to target the different audio protocols with their own specificities concerning this Frame synchronization behavior. This reconfigurability is done using register SAI\_xFRCR. [Figure 378](#) illustrates this flexibility.

**Figure 378. Audio frame**



In AC'97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI\_xCR1 register), the frame synchronization shape is forced to match the AC'97 protocol. The SAI\_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

#### Frame length

- Master mode

The audio frame length can be configured to up to 256 bit clock cycles, by setting FRL[7:0] field in the SAI\_xFRCR register.

If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit is extended to 0 or the SD line is released to Hi-z depending

the state of bit TRIS in the SAI\_xCR2 register (refer to [FS signal role](#)). In reception mode, the remaining bit is ignored.

If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.

If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to [Section 36.4.8: SAI clock generator](#).

- Slave mode

The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the Frame synchronization signal during an ongoing audio frame. In this case an error is generated. For more details refer to [Section 36.4.14: Error flags](#).

In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI\_xFRCR register.

The number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame is 8.

### Frame synchronization polarity

FSPOL bit in the SAI\_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. Start of frame is synchronized to this signal. It is effective only if the start of frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to [Section 36.4.14: Error flags](#)).

In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI\_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition is managed as described in [Section 36.4.14: Error flags](#), but the audio communication flow is not interrupted.

### Frame synchronization active level length

The FSALL[6:0] bits of the SAI\_xFRCR register enable the configuration of the length of the active level of the Frame synchronization signal. The length can be set from 1 to 128 bit clock cycles.

As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM.

### Frame synchronization offset

Depending on the audio protocol targeted in the application, the Frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). FSOFF bit in the SAI\_xFRCR register enables to choose one of the two configurations.

### FS signal role

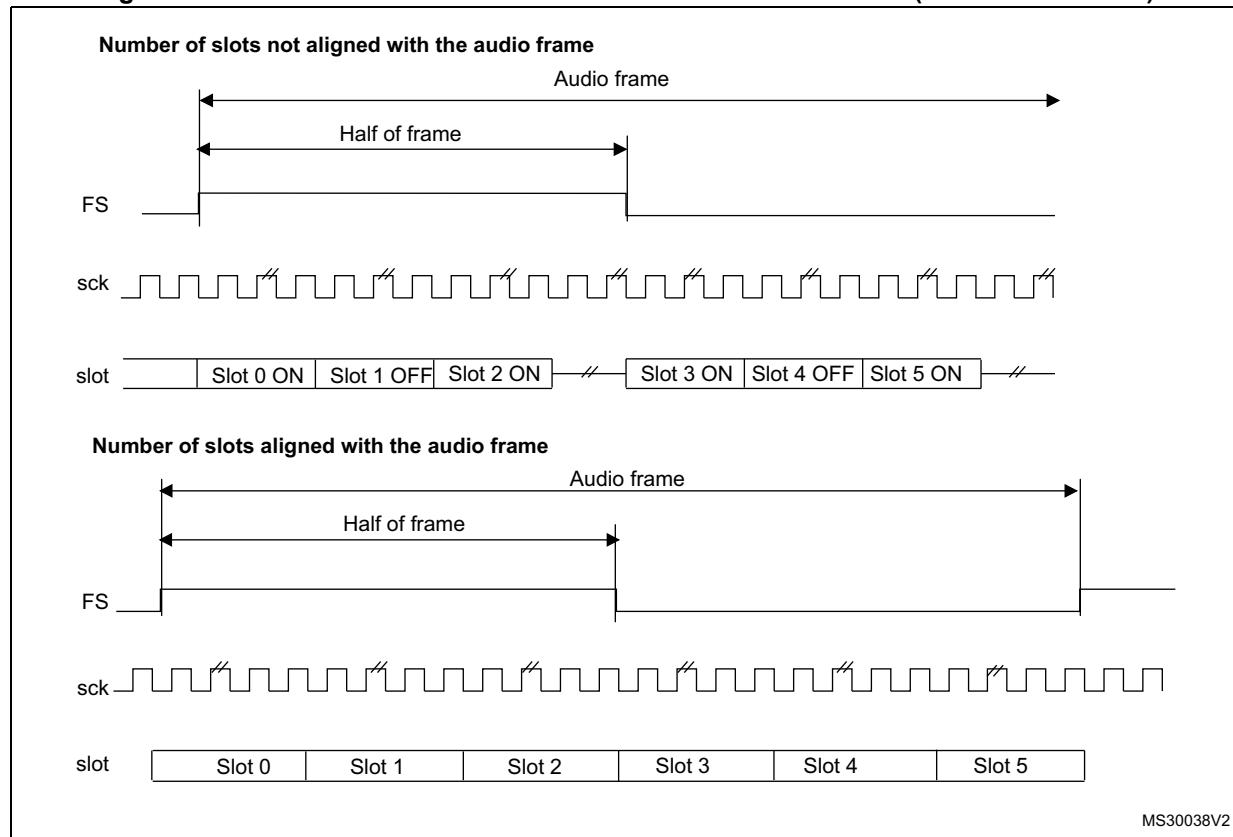
The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI\_xFRCR register selects which meaning it has:

- 0: start of frame, like for instance the PCM/DSP, TDM, AC'97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S, the MSB or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI\_xCR2 register.

Otherwise if TRIS = 1, the SD line is released to HI-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

**Figure 379. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)**

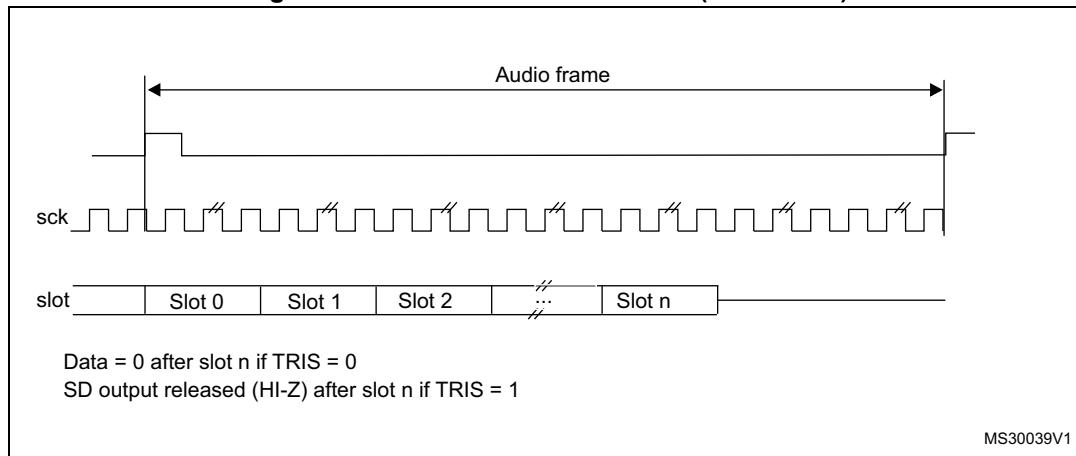


1. The frame length should be even.

If FSDEF bit in SAI\_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI\_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI\_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI\_xFRCR register), then:

- if TRIS = 0 in the SAI\_xCR2 register, the remaining bit after the last slot is forced to 0 until the end of frame in case of transmitter,
- if TRIS = 1, the line is released to HI-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

Figure 380. FS role is start of frame (FSDEF = 0)



The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O is released and left free for other purposes.

### 36.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

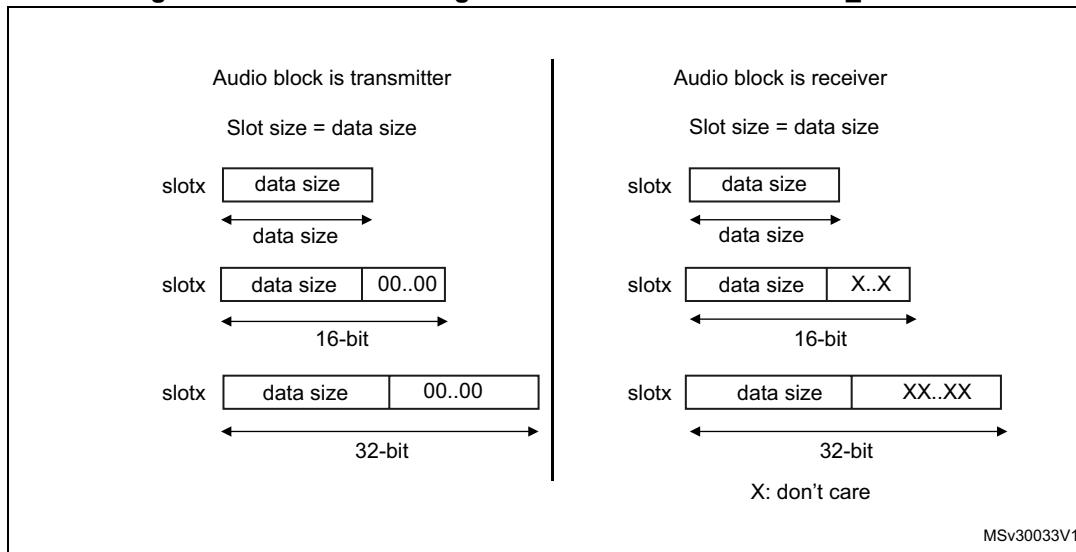
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI\_xSLOTR register.

When an invalid slot is transferred, the SD data line is either forced to 0 or released to HI-Z depending on TRIS bit configuration (refer to [Output data line management on an inactive slot](#)) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there is no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

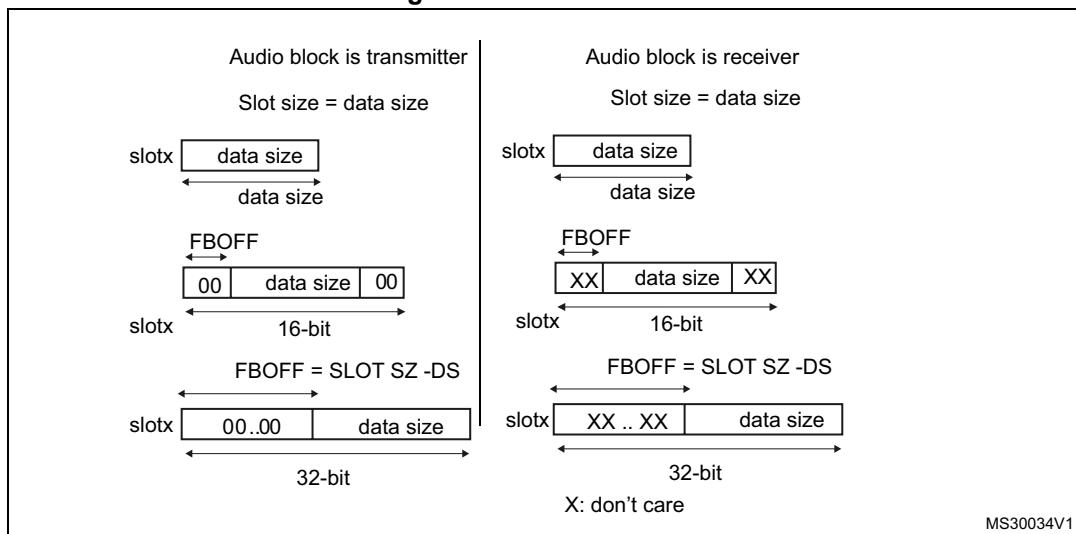
The slot size is also configurable as shown in [Figure 381](#). The size of the slots is selected by setting SLOTSZ[1:0] bits in the SAI\_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 381. Slot size configuration with FBOFF = 0 in SAI\_xSLOTR



It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI\_xSLOTR register. 0 values are injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

Figure 382. First bit offset



It is mandatory to respect the following conditions to avoid bad SAI behavior:

FBOFF  $\leq$  (SLOTSZ - DS),

DS  $\leq$  SLOTSZ,

NBSLOT  $\times$  SLOTSZ  $\leq$  FRL (frame length),

The number of slots must be even when bit FSDEF in the SAI\_xFRCR register is set.

In AC'97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the slot size is automatically set as defined in [Section 36.4.11: AC'97 link controller](#).

### 36.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK\_x) and bit clock (SCK\_x) signals from the sai\_x\_ker\_ck. The sai\_x\_ker\_ck clock is delivered by the clock controller of the product (RCC).

#### Generation of the master clock (MCLK\_x)

The clock generator provides the master clock (MCLK\_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK\_x clock is used as system clock for an external audio device, since it enables the generation of the MCLK\_x before activating the audio stream.

To generate a master clock on MCLK\_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK\_x output, the following operations are not recommended:

- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI guarantees that there is no spurs on MCLK\_x output when the MCLK\_x is switched ON and OFF via MCKEN bit (with SAIEN = 0).

*Table 226* shows MCLK\_x activation conditions.

**Table 226. MCLK\_x activation conditions**

MCLKEN	NODIV	SAIEN for block x	MCLK_x
0	X	0	Disabled
1			Enabled
0	1	1	Disabled
1			Enabled
X	0		Enabled

*Note:* MCLK\_x can also be generated in AC'97 mode, when MCLKEN is set to 1.

### Generation of the bit clock (SCK\_x)

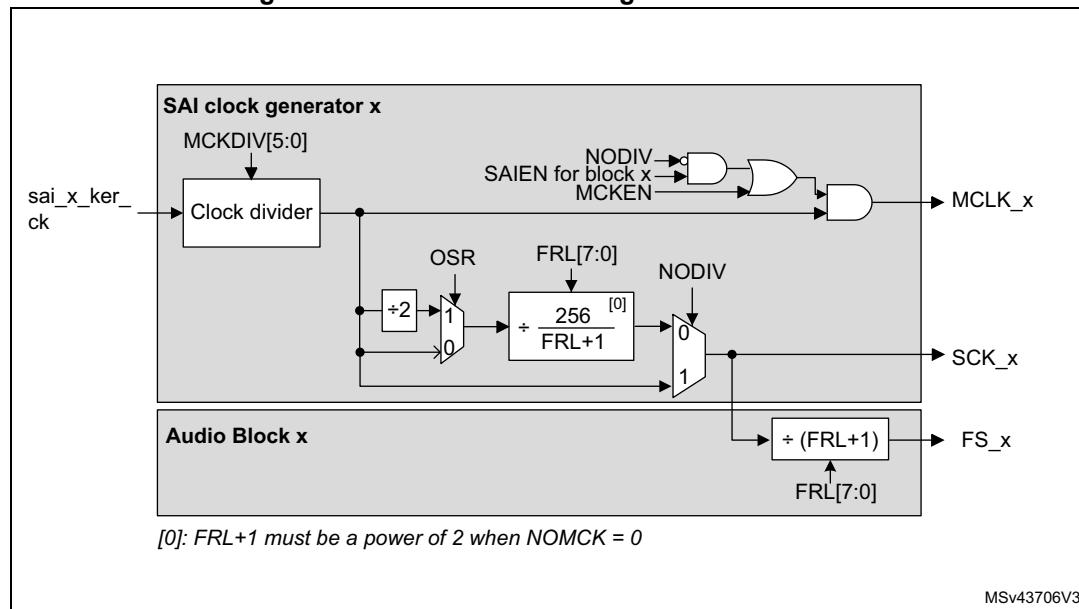
The clock generator provides the bit clock (SCK\_x) when the audio block is defined as Master. The frame synchronization (FS\_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK\_x clock is not generated.

The bit clock strobing edge of SCK\_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

*Figure 383* illustrates the architecture of the audio block clock generator.

**Figure 383. Audio block clock generator overview**



The NODIV bit must be used to force the ratio between the master clock (MCLK\_x) and the frame synchronization (FS\_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given hereafter.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK\_x) via MCKDIV. In addition there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (i.e. FRL[7:0] > 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK\_x is equal to the SCK\_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR and MCKDIV[5:0] bits belong to the SAI\_xCR1 register, while FRL[7:0] belongs to SAI\_xFRCR.

### Clock generator programming when NODIV = 0

In that case, MCLK\_x frequency is:

- $F_{MCLK\_x} = 256 \times F_{FS\_x}$  if OSR = 0
- $F_{MCLK\_x} = 512 \times F_{FS\_x}$  if OSR = 1

When MCKDIV is different from 0, MCLK\_x frequency is given by the formula below:

$$F_{MCLK\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

The frame synchronization frequency is given by:

$$F_{FS\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV \times (OSR + 1) \times 256}$$

The bit clock frequency (SCK\_x) is given by the following formula:

$$F_{SCK\_x} = \frac{F_{sai\_x\_ker\_ck} \times (FRL + 1)}{MCKDIV \times (OSR + 1) \times 256}$$

**Note:** When NODIV is equal to 0, (FRL+1) must be a power of two. In addition (FRL+1) must range between 8 and 256. (FRL + 1) represents the number of bit clock in the audio frame.

When MCKDIV division ratio is odd, the MCLK duty cycle is not 50%. The bit clock signal (SCK\_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if  $(FRL+1) = 2^8$ .

It is recommended, to program MCKDIV to an even value or to big values (higher than 10).

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

### Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK\_x) is given in the formula below:

$$F_{SCK\_x} = F_{MCLK\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

The frequency of the frame synchronization (FS\_x) is given by the following formula:

$$F_{FS\_x} = \frac{F_{sai\_x\_ker\_ck}}{(FRL + 1) \times MCKDIV}$$

**Note:** When NODIV is set to 1, (FRL+1) can take any values from 8 to 256.

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

### Clock generator programming examples

*Table 227* gives programming examples for 48, 96 and 192 kHz.

**Table 227. Clock generator programming examples**

Input sai_x_ker_ck clock frequency	MCLK	$F_{MCLK}/F_{FS}$	$F_{RL}^{(1)}$	OSR	NODIV	MCKEN	$MCKDIV[5:0]$	Audio Sampling frequency ( $F_{FS}$ )
98.304 MHz	Y	512	$2^{N-1}$	1	0	1	0 or 1	192 kHz
		512	$2^{N-1}$	1	0	1	2	96 kHz
		512	$2^{N-1}$	1	0	1	4	48 kHz
		256	$2^{N-1}$	0	0	1	2	192 kHz
		256	$2^{N-1}$	0	0	1	4	96 kHz
		256	$2^{N-1}$	0	0	1	8	48 kHz
	N	-	63	-	1	0	8	192 kHz
		-	63	-	1	0	16	96 kHz
		-	63	-	1	0	32	48 kHz

1. N is an integer value between 3 and 8.

#### 36.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. There is therefore only one FIFO request linked to FREQ bit in the SAI\_xSR register.

An interrupt is generated if FREQIE bit is enabled in the SAI\_xIM register. This depends on:

- FIFO threshold setting (FLVL bits in SAI\_xCR2)
- Communication direction (transmitter or receiver). Refer to *Interrupt generation in transmitter mode* and *Interrupt generation in reception mode*.

##### Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:

- When the FIFO threshold bits in SAI\_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in SAI\_xSR register) if no data are available in SAI\_xDR register (FLVL[2:0] bits in SAI\_xSR is less than 001b). This Interrupt (FREQ bit in SAI\_xSR register) is cleared by hardware when the FIFO is no more empty (FLVL[2:0] bits in SAI\_xSR are different from 0b000) i.e one or more data are stored in the FIFO.
- When the FIFO threshold bits in SAI\_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 001b), an interrupt is generated (FREQ bit set by hardware to 1 in SAI\_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI\_xSR are less than 0b010). This Interrupt (FREQ bit in SAI\_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI\_xSR are higher or equal to 0b010).
- When the FIFO threshold bits in SAI\_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in

SAI\_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI\_xSR are less than 011b). This Interrupt (FREQ bit in SAI\_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI\_xSR are higher or equal to 011b).

- When the FIFO threshold bits in SAI\_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 011b), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI\_xSR register) if less than three quarters of the FIFO contain data (FLVL[2:0] bits in SAI\_xSR are less than 0b100). This Interrupt (FREQ bit in SAI\_xSR register) is cleared by hardware when at least three quarters of the FIFO contain data (FLVL[2:0] bits in SAI\_xSR are higher or equal to 0b100).
- When the FIFO threshold bits in SAI\_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI\_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI\_xSR is less than 101b). This Interrupt (FREQ bit in SAI\_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI\_xSR is equal to 101b value).

### Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in SAI\_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI\_xSR register) if at least one data is available in SAI\_xDR register(FLVL[2:0] bits in SAI\_xSR is higher or equal to 001b). This Interrupt (FREQ bit in SAI\_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in SAI\_xSR is equal to 0b000) i.e no data are stored in FIFO.
- When the FIFO threshold bits in SAI\_xCR2 register are configured as FIFO quarter fully (FTH[2:0] set to 001b), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI\_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI\_xSR is higher or equal to 0b010). This Interrupt (FREQ bit in SAI\_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI\_xSR is less than 0b010).
- When the FIFO threshold bits in SAI\_xCR2 register are configured as FIFO half fully (FTH[2:0] set to 0b010 value), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI\_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI\_xSR is higher or equal to 011b). This Interrupt (FREQ bit in SAI\_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI\_xSR is less than 011b).
- When the FIFO threshold bits in SAI\_xCR2 register are configured as FIFO three quarter full(FTH[2:0] set to 011b value), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI\_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI\_xSR is higher or equal to 0b100). This Interrupt (FREQ bit in SAI\_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available(FLVL[2:0] bits in SAI\_xSR is less than 0b100).
- When the FIFO threshold bits in SAI\_xCR2 register are configured as FIFO full(FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI\_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI\_xSR is equal to 101b). This Interrupt (FREQ bit in SAI\_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI\_xSR is less than 101b).

Like interrupt generation, the SAI can use the DMA if DMAEN bit in the SAI\_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI\_xDR register.

Data should be right aligned when it is written in the SAI\_xDR.

Data received are right aligned in the SAI\_xDR.

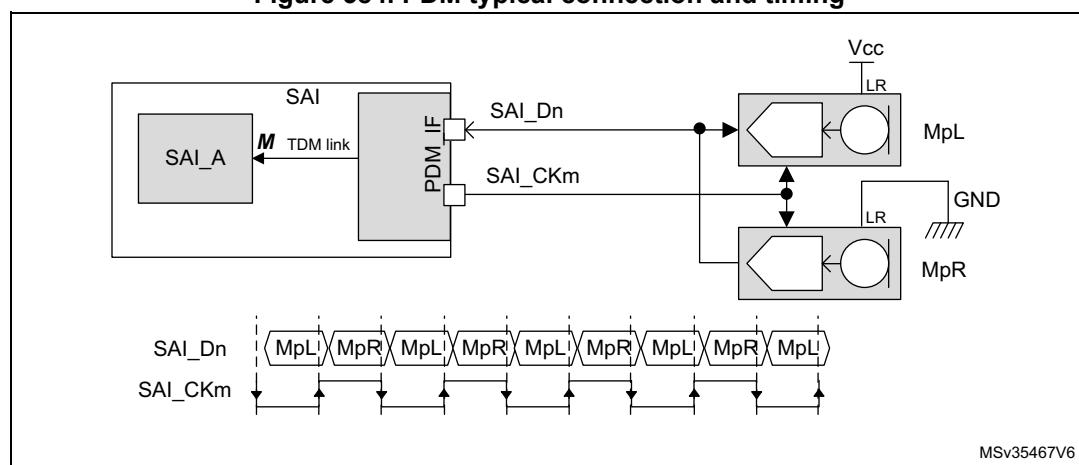
The FIFO pointers can be reinitialized when the SAI is disabled by setting bit FFLUSH in the SAI\_xCR2 register. If FFLUSH is set when the SAI is enabled the data present in the FIFO are lost automatically.

### 36.4.10 PDM interface

The PDM (Pulse Density Modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. Depending on product implementation, less microphones can be supported (refer to [Section 36.3: SAI implementation](#)).

[Figure 384](#) shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI\_CK[m] rising edge while the other provides valid data on SAI\_CK[m] falling edge (m being the number of clock lines).

**Figure 384. PDM typical connection and timing**



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI\_A subblock configured in TDM master mode. It cannot be used with SAI\_B subblock. The PDM interface uses the timing signals provided by the TDM interface of SAI\_A and adapts them to generate a bitstream clock (SAI\_CK[m]).

The data processing sequence into the PDM is the following:

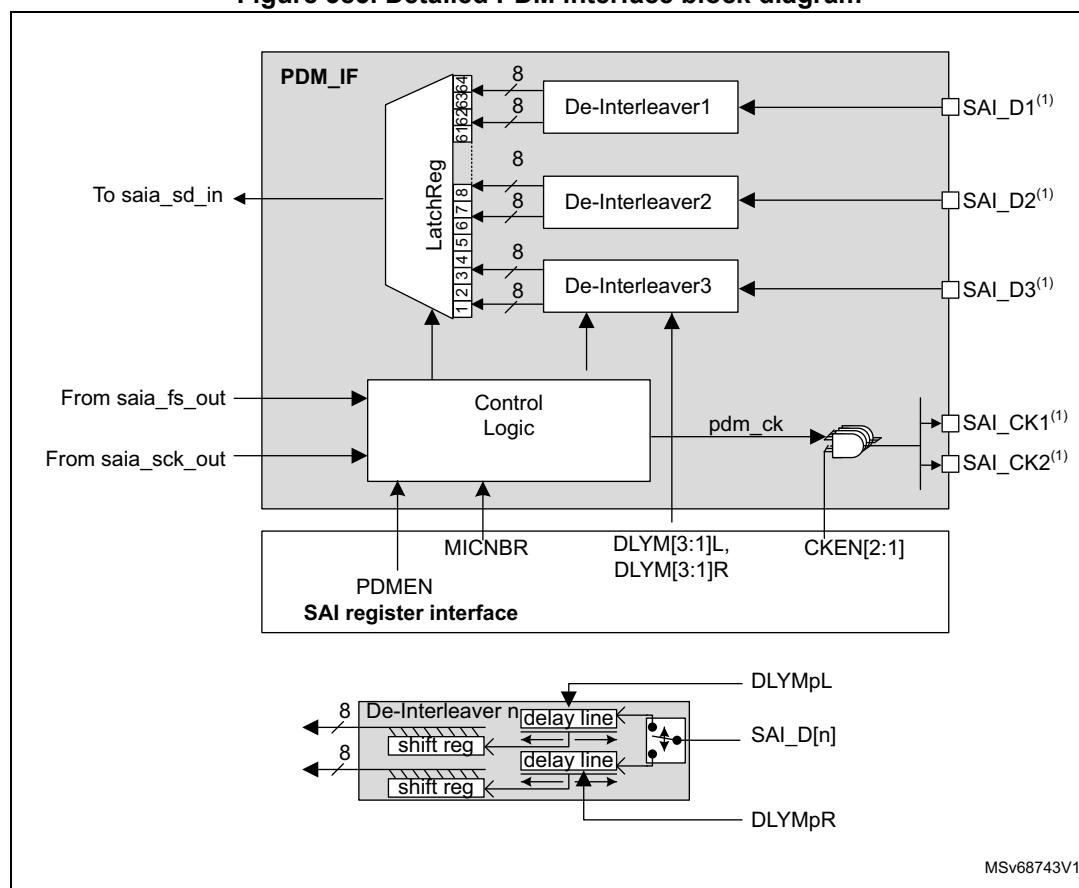
1. The PDM interface builds the bitstream clock from the bit clock received from the TDM interface of SAI\_A.
2. The bitstream data received from the microphones (SAI\_D[n]) are de-interleaved and go through a 7-bit delay line in order to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI\_A via the serial data line of the TDM interface.

*Figure 385* hereafter shows the block diagram of PDM interface, with a detailed view of a de-interleaver.

**Note:**

*The PDM interface does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.*

**Figure 385. Detailed PDM interface block diagram**



1. These signals might not be available in all SAI instances. Refer to [Section 36.3: SAI implementation](#) for details.
2. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM interface can be enabled through the PDMEN bit in SAI\_PDMCR register. However the PDM interface must be enabled prior to enabling SAI\_A block.

To reduce the memory footprint, the user can select the amount of microphones the application needs. This can be done through MICNBR[1:0] bits. It is possible to select

between 2,4,6 or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

### Enabling the PDM interface

To enable the PDM interface, follow the sequence below:

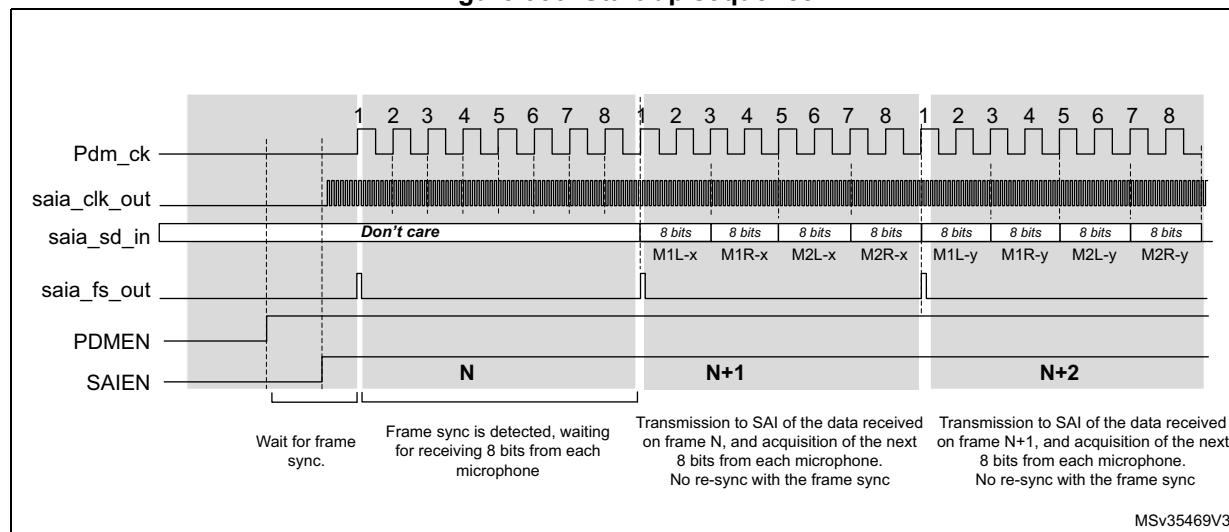
1. Configure SAI\_A in TDM master mode (see [Table 228](#)).
2. Configure the PDM interface as follows:
  - a) Define the number of digital microphones via MICNBR.
  - b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via PDMEN bit.
4. Enable the SAI\_A.

**Note:** *Once the PDM interface and SAI\_A are enabled, the first 2 TDMA frames received on SAI\_ADR are invalid and must be dropped.*

### Start-up sequence

[Figure 386](#) shows the start-up sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI\_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the TDM interface.

**Figure 386. Start-up sequence**



### SAI\_ADR data format

The arrangement of the data coming from the microphone into the SAI\_ADR register depends on the following parameters:

- The amount of microphones
- The slot width selected
- LSBFIRST bit.

The slot width defines the amount of significant bits into each word available into the SAI\_ADR.

When a slot width of 32 bits is selected, each data available into the SAI\_ADR contains 32 useful bits. This reduces the amount of words stored into the memory. However the counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

In the other hand, when the slot width is set to 8 bits, each data available into the SAI\_ADR contain 8 useful bits. This increases the amount of words stored into the memory. However, it offers the advantage to avoid extra processing since each word contains information from one microphone.

#### SAI\_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to [Figure 387](#).

For an 8 microphone configuration, two consecutive words read from the SAI\_ADR register contain a data byte from each microphone.

For a 4 microphones configuration, each word read from the SAI\_ADR register contains a data byte from each microphone.

**Figure 387. SAI\_ADR format in TDM, 32-bit slot width**

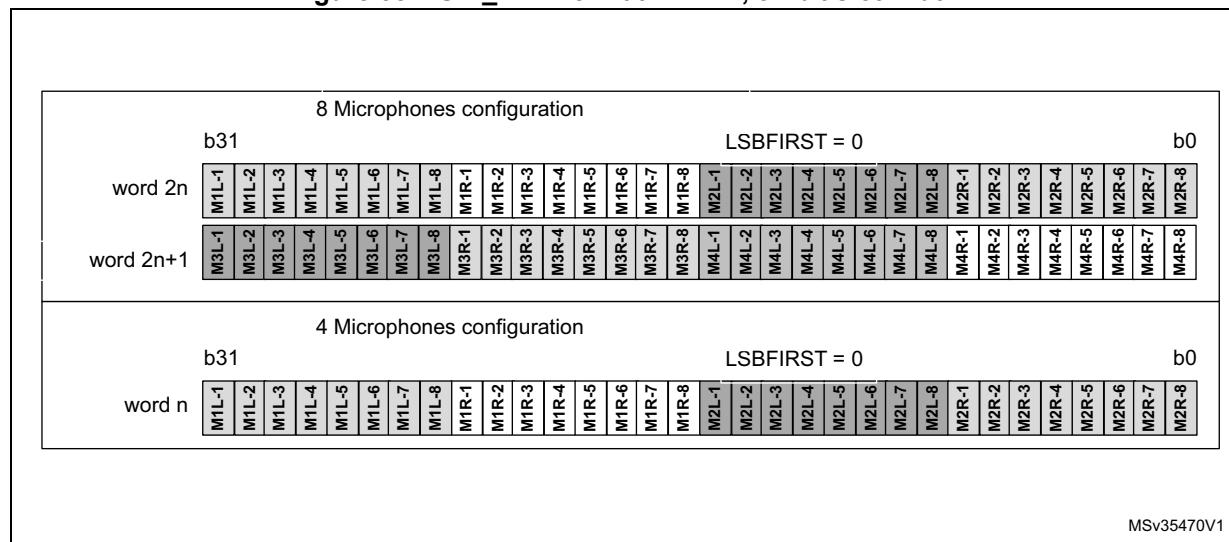
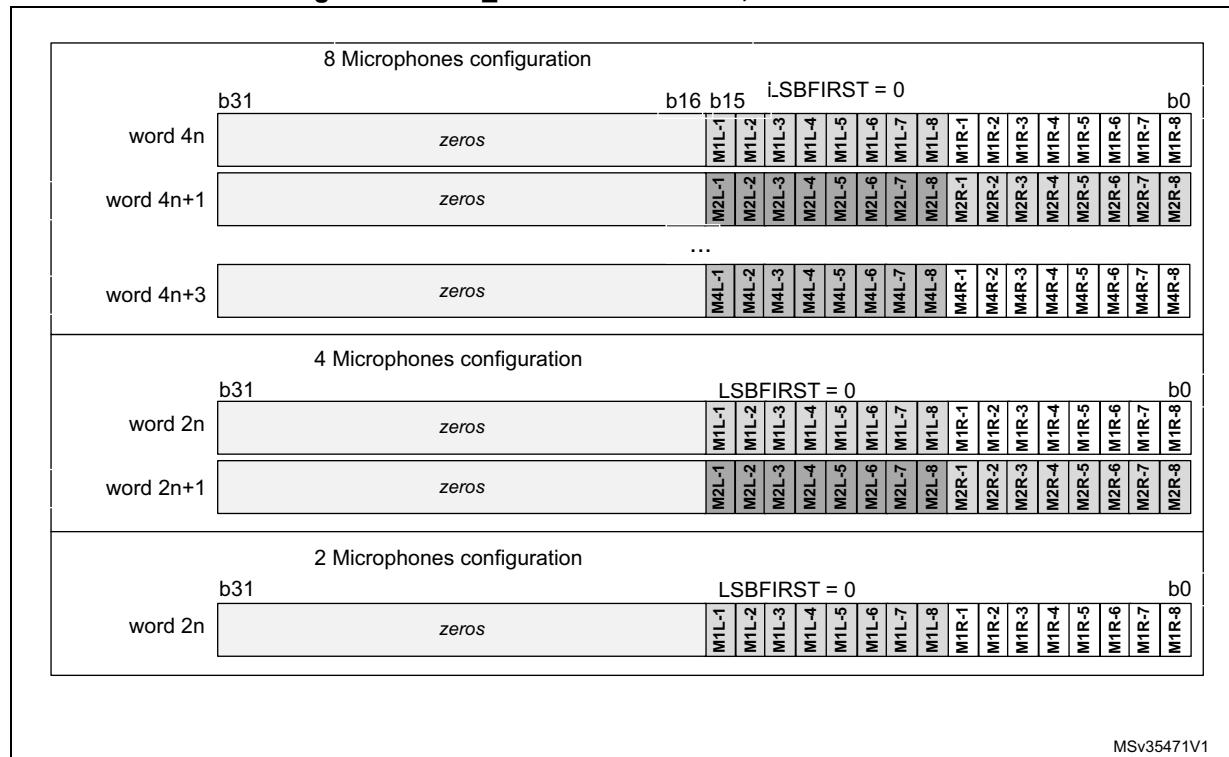


Figure 388. SAI\_ADR format in TDM, 16-bit slot width

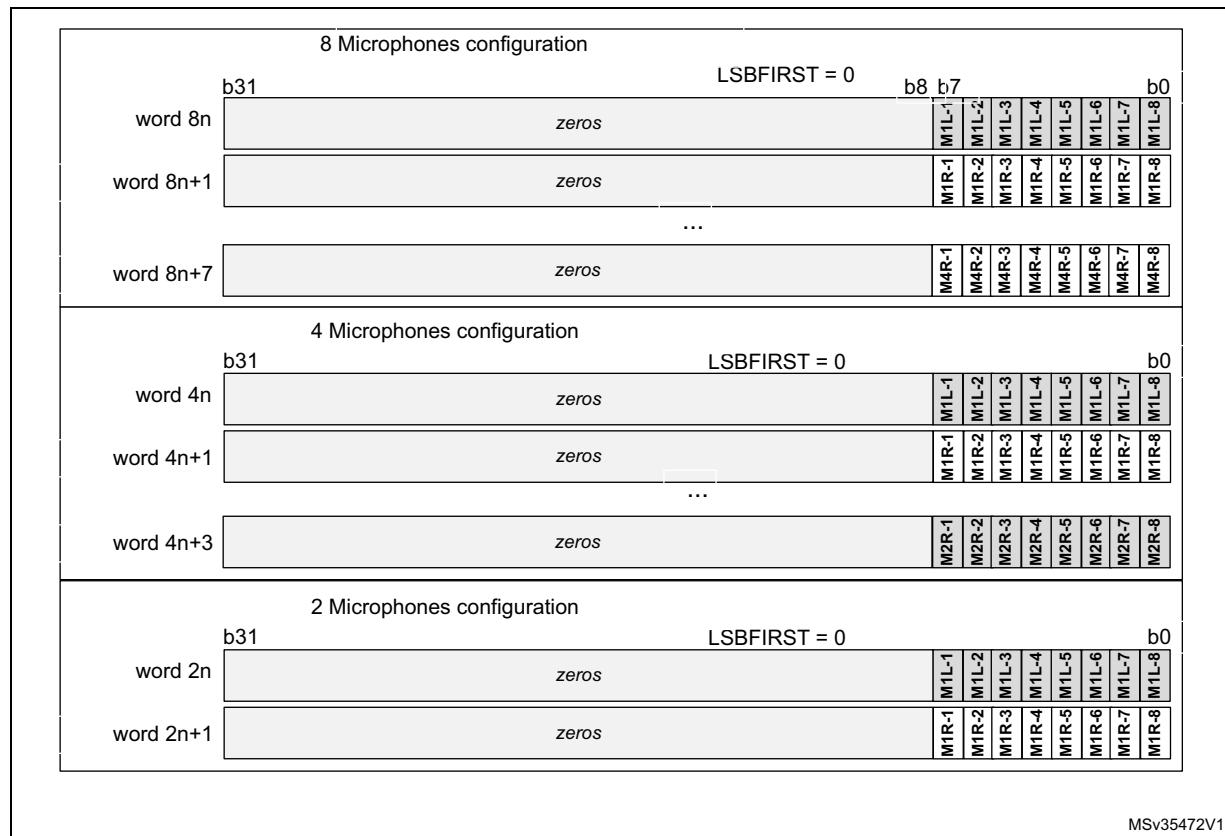


- **Using a 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to [Figure 389](#).

For an 8 microphone configuration, 8 consecutive words read from the SAI\_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI\_ADR are right aligned.

For 4 or 2 microphone configuration, the SAI behavior is similar to 8 microphone configurations. Up to 4 words of 8 bits are required to acquire a byte from 4 microphones and 2 words from 2 microphones.

Figure 389. SAI\_ADR format in TDM, 8-bit slot width



### TDM configuration for PDM interface

SAI\_A TDM interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the PDM interface as shown in [Table 228](#) to ensure a good connection with the PDM interface.

Table 228. TDM settings

Bit Fields	Values	Comments
MODE	0b01	Mode must be MASTER receiver
PRTCFIG	0b00	Free protocol for TDM
DS	X	To be adjusted according to the required data format, in accordance to the frame length and the number of slots (FRL and NBSLOT). See <a href="#">Table 229</a> .
LSBFIRST	X	This parameter can be used according to the wanted data format
CKSTR	0	Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.
MONO	0	Stereo mode
FRL	X	To be adjusted according to the number of microphones (MICNBR). See <a href="#">Table 229</a> .
FSALL	0	Pulse width is one bit clock cycle
FSDEF	0	FS signal is a start of frame

Table 228. TDM settings (continued)

Bit Fields	Values	Comments
FSPOL	1	FS is active High
FSOFF	0	FS is asserted on the first bit of slot 0
FBOFF	0	No offset on slot
SLOTSZ	0	Slot size = data size
NBSLOT	X	To be adjusted according to the required data format, in accordance to the slot size, and the frame length (FRL and DS). See <a href="#">Table 229</a> .
SLOTEN	X	To be adjusted according to NBSLOT
NODIV	1	No need to generate a master clock MCLK
MCKDIV	X	Depends on the frequency provided to sai_a_ker_ck input. This parameter must be adjusted to generate the proper bitstream clock frequency. See <a href="#">Table 229</a> .

### Adjusting the bitstream clock rate

To properly program the SAI TDM interface, the user application must take into account the settings given in [Table 228](#), and follow the below sequence:

1. Adjust the bit clock frequency ( $F_{SCK\_A}$ ) according to the required frequency for the PDM bitstream clock, using the following formula:

$$F_{SCK\_A} = F_{PDM\_CK} \times (MICNBR + 1) \times 2$$

MICNBR can be 0,1,2 or 3 (0 = 2 microphones., see [Section 36.6.17](#))

2. Set the frame length (FRL) using the following formula

$$FRL = (16 \times (MICNBR + 1)) - 1$$

3. Configure the slot size (DS) to a multiple of (FRL+1).

Table 229. TDM frame configuration examples<sup>(1)(2)</sup>

Microphone sampling rate	Nber of microphones	Wanted SAI_CKn frequency	bit clock (SCK_A) frequency	Frame sync. (FS_A) frequency	FRL	DS	NBSLOT	Comments
48 kHz	up to 8	3.072 MHz	24.576 MHz	384 kHz	63	0b111	1	2 slots of 32 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b100	3	4 slots of 16 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	3.072 MHz	18.432 MHz	384 kHz	47	0b110	1	2 slots of 24 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b100	2	3 slots of 16 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	3.072 MHz	12.288 MHz	384 kHz	31	0b111	0	1 slot of 32 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b100	1	2 slots of 16 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	3.072 MHz	6.144 MHz	384 kHz	15	0b100	0	1 slots of 16 bits per frame
		3.072 MHz	6.144 MHz	384 kHz	15	0b010	1	2 slots of 8 bits per frame
16 kHz	up to 8	1.024 MHz	8.192 MHz	128 kHz	63	0b111	1	2 slots of 32 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b100	3	4 slots of 16 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	1.024 MHz	6.144 MHz	128 kHz	47	0b110	1	2 slots of 24 bits per frame
		1.024 MHz	6.144 MHz	128 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	1.024 MHz	4.096 MHz	128 kHz	31	0b111	0	1 slot of 32 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b100	1	2 slots of 16 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	1.024 MHz	2.048 MHz	128 kHz	15	0b100	0	1 slot of 16 bits per frame
		1.024 MHz	2.048 MHz	128 kHz	15	0b010	1	2 slots of 8 bits per frame

1. Refer to [Table 228: TDM settings](#) for additional information on TDM configuration. The sai\_a\_ker\_ck clock frequency provided to the SAI must be a multiple of the SCK\_A frequency, and MCKDIV should be programmed accordingly.
2. The above sai\_a\_ker\_ck frequencies are given as examples only. Refer to section *Reset and clock controller (RCC) to check if they can be generated on the device*.
3. The table above gives allowed settings for a decimation ratio of 64.

### Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via SAI\_PDMDLY register.

The new delays values become effective after two TDM frames.

### 36.4.11 AC'97 link controller

The SAI is able to work as an AC'97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

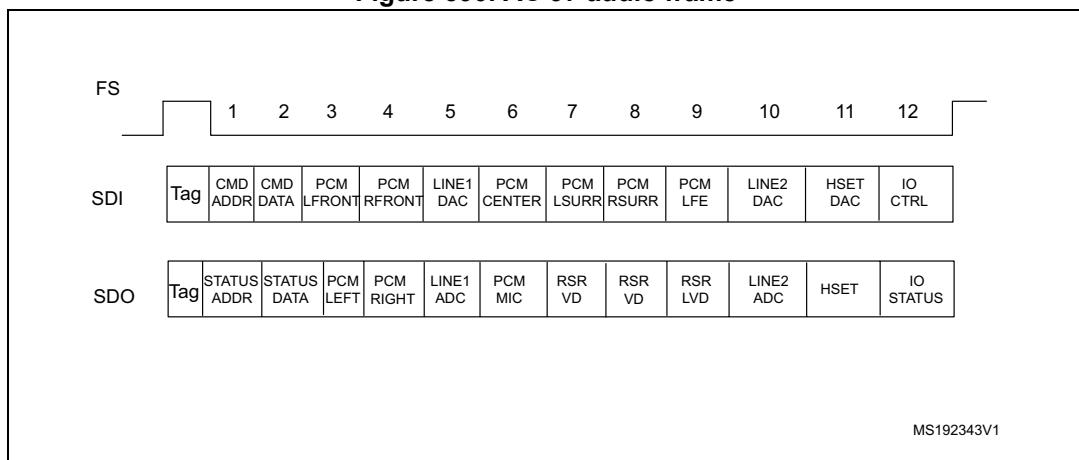
To select this protocol, set PRTC<sub>FG</sub>[1:0] bits in the SAI\_xCR1 register to 10. When AC'97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed to 13 slots. The first one is 16-bit wide and all the others are 20-bit wide (data slots).
- FBOFF[4:0] bits in the SAI\_xSLOTR register are ignored.
- The SAI\_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC'97 controller link drives the FS signal whatever the master or slave configuration.

*Figure 390* shows an AC'97 audio frame structure.

**Figure 390. AC'97 audio frame**



**Note:** In AC'97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC'97 protocol standard.

One SAI can be used to target an AC'97 point-to-point communication.

In receiver mode, the SAI acting as an AC'97 link controller requires no FIFO request and so no data storage in the FIFO when the Codec ready bit in the slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI\_xIM register, flag CNRDY is set in the SAI\_xSR register and an interrupt is generated. This flag is dedicated to the AC'97 protocol.

### Clock generator programming in AC'97 mode

In AC'97 mode, the frame length is fixed at 256 bits, and its frequency must be set to 48 kHz. The formulas given in [Section 36.4.8: SAI clock generator](#) must be used with  $F_{RL} = 255$ , in order to generate the proper frame rate ( $F_{FS\_x}$ ).

#### 36.4.12 SPDIF output

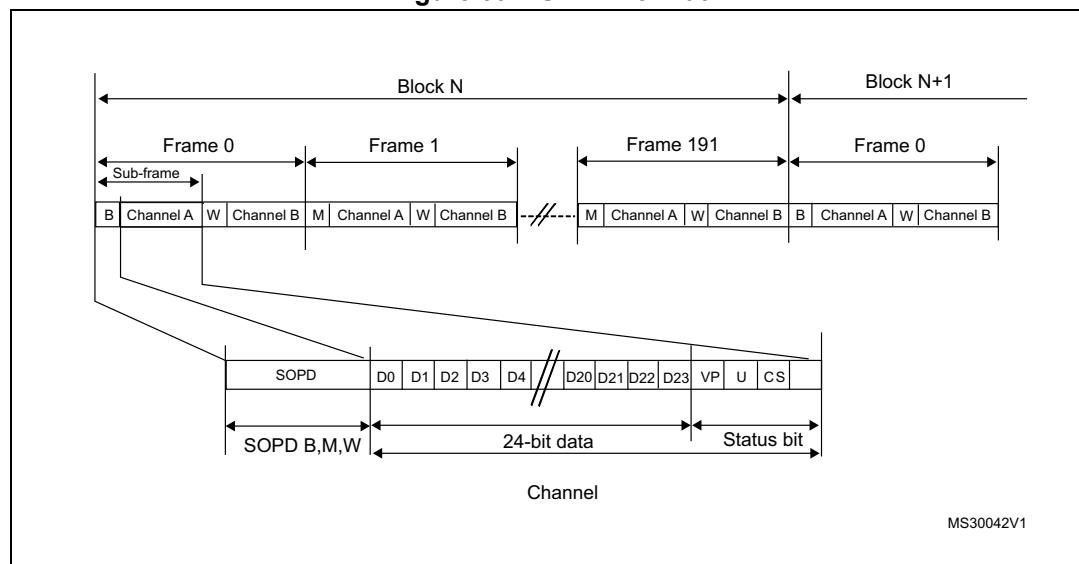
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set PRTC<sub>FG</sub>[1:0] bit to 01 in the SAI\_xCR1 register.

For SPDIF protocol:

- Only SD data line is enabled.
- FS, SCK, MCLK I/Os pins are left free.
- MODE[1] bit is forced to 0 to select the master mode in order to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in DS[2:0] bits in the SAI\_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock should be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI\_xFRCR and SAI\_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in [Figure 391](#).

**Figure 391. SPDIF format**



A SPDIF block contains 192 frames. Each frame is composed of two 32-bit sub-frames, generally one for the left channel and one for the right channel. Each sub-frame is composed of a SOPD pattern (4-bit) to specify if the sub-frame is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see [Table 230](#)). The next 28 bits of channel information are composed of 24 bits data + 4 status bits.

Table 230. SOPD pattern

SOPD	Preamble coding		Description
	last bit is 0	last bit is 1	
B	11101000	00010111	Channel A data at the start of block
W	11100100	00011011	Channel B data somewhere in the block
M	11100010	00011101	Channel A data

The data stored in SAI\_xDR has to be filled as follows:

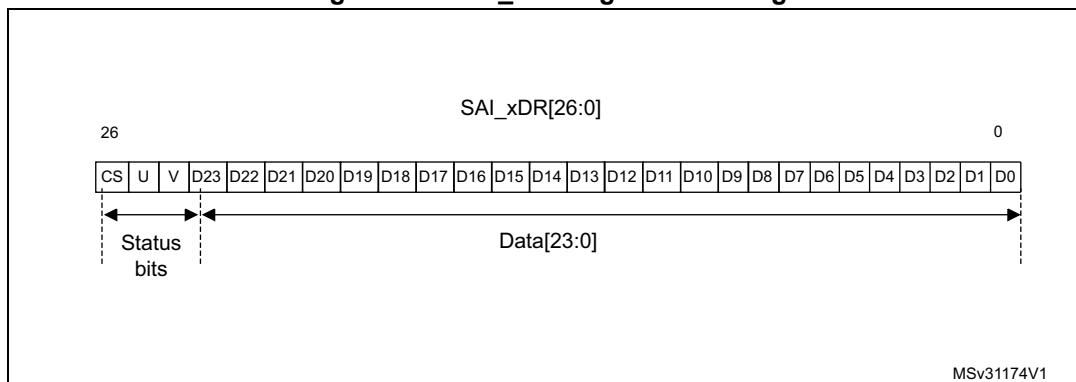
- SAI\_xDR[26:24] contain the Channel status, User and Validity bits.
- SAI\_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, then data must be mapped on SAI\_xDR[23:4].

If the data size is 16 bits, then data must be mapped on SAI\_xDR[23:8].

SAI\_xDR[23] always represents the MSB.

Figure 392. SAI\_xDR register ordering



Note:

*The transfer is performed always with LSB first.*

The SAI first sends the adequate preamble for each sub-frame in a block. The SAI\_xDR is then sent on the SD line (manchester coded). The SAI ends the sub-frame by transferring the Parity bit calculated as described in [Table 231](#).

Table 231. Parity bit calculation

SAI_xDR[26:0]	Parity bit P value transferred
odd number of 0	0
odd number of 1	1

The underrun is the only error flag available in the SAI\_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence should be

executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in SAI\_xCR1 register.
3. Clear the COVRUNDR flag in the SAI\_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI\_xCR2.

The software needs to point to the address of the future data corresponding to a start of new block (data for preamble B). If the DMA is used, the DMA source base address pointer should be updated accordingly.

5. Enable again the DMA stream (DMA peripheral) if the DMA used to manage data transfers according to the new source base address.
6. Enable again the SAI by setting SAIEN bit in SAI\_xCR1 register.

### Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI provides a bit clock twice faster as the symbol-rate. The table hereafter shows usual examples of symbol rates with respect to the audio sampling rate.

**Table 232. Audio sampling frequency versus symbol rates**

Audio sampling frequencies ( $F_S$ )	Symbol-rate
44.1 kHz	2.8224 MHz
48 kHz	3.072 MHz
96 kHz	6.144 MHz
192 kHz	12.288 MHz

More generally, the relationship between the audio sampling frequency ( $F_S$ ) and the bit clock rate ( $F_{SCK\_x}$ ) is given by the formula:

$$F_S = \frac{F_{SCK\_x}}{128}$$

The bit clock rate is obtained as follows:

$$F_{SCK\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

**Note:** *The above formulas are valid only if NODIV is set to 1 in SAI\_ACR1 register.*

### 36.4.13 Specific features

The SAI interface embeds specific features which can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI\_xCR2 register.

#### Mute mode

The mute mode can be used when the audio subblock is a transmitter or a receiver.

##### Audio subblock in transmission mode

In transmitter mode, the mute mode can be selected at anytime. The mute mode is active for entire audio frames. The MUTE bit in the SAI\_xCR2 register enables the mute mode when it is set during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame is still a mute frame.

If the number of slots set through NBSLOT[3:0] bits in the SAI\_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via MUTEVAL bit in the SAI\_xCR2 register.

If the number of slots set in NBSLOT[3:0] bits in the SAI\_xSLOTR register is greater than 2, MUTEVAL bit in the SAI\_xCR2 is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

##### Audio subblock in reception mode

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI\_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI\_xSR register is set and an interrupt can be generated if MUTEDETIE bit is set in SAI\_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

*Note:* The mute mode is not available for SPDIF audio blocks.

#### Mono/stereo mode

In transmitter mode, the mono mode can be addressed, without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI\_xSLOTR). In this case, the access time to and from the FIFO is reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode,

1. Set MONO bit to 1 in the SAI\_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI\_xSLOTR.

In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2 as in transmitter mode. When it is set, only slot 0 data are stored in the FIFO. The data belonging to slot 1 are discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

### Companding mode

Telecommunication applications can require to process the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI\_xCR2 register (used only when Free protocol mode is selected), the application software can choose to process or not the data before sending it on SD serial output line (compression) or to expand the data after the reception on SD serial input line (expansion) as illustrated in [Figure 393](#). The two companding modes supported are the  $\mu$ -Law and the A-Law log which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the  $\mu$ -Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI\_xCR2 register).

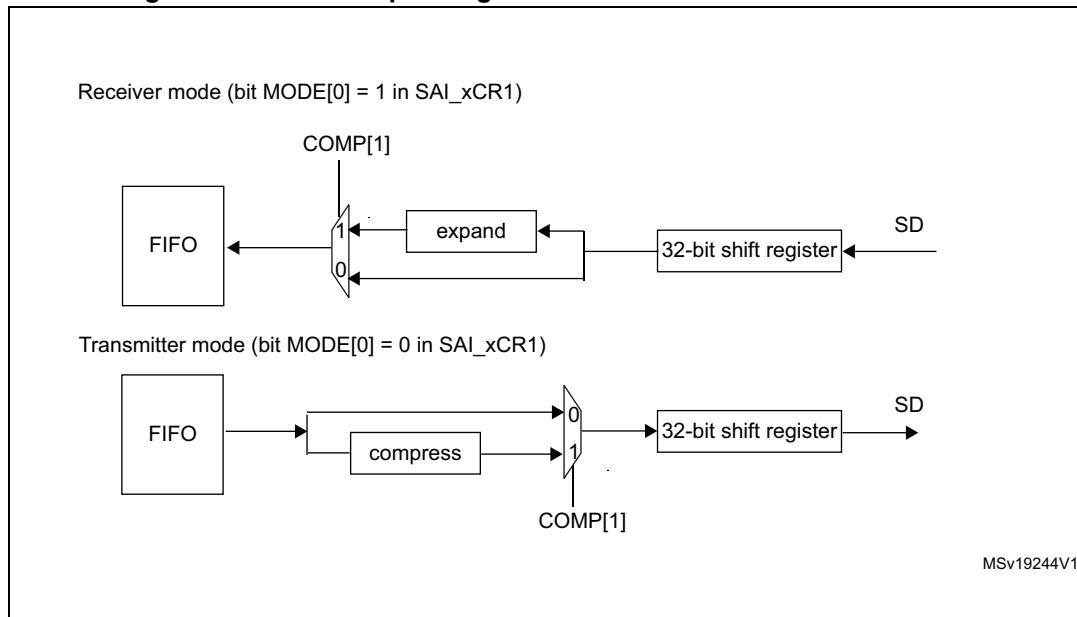
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI\_xCR2 register).

Both  $\mu$ -Law or A-Law companding standard can be computed based on 1's complement or 2's complement representation depending on the CPL bit setting in the SAI\_xCR2 register.

In  $\mu$ -Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8-bit wide. For this reason, DS[2:0] bits in the SAI\_xCR1 register are forced to 010 when the SAI audio block is enabled (SAIEN bit = 1 in the SAI\_xCR1 register) and when one of these two companding modes selected through the COMP[1:0] bits.

If no companding processing is required, COMP[1:0] bits should be kept clear.

Figure 393. Data companding hardware in an audio block in the SAI



1. Not applicable when AC'97 or SPDIF are selected.

Expansion and compression mode are automatically selected through the SAI\_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI\_xCR2 register, the compression mode is applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm is applied.

#### Output data line management on an inactive slot

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via TRIS bit).

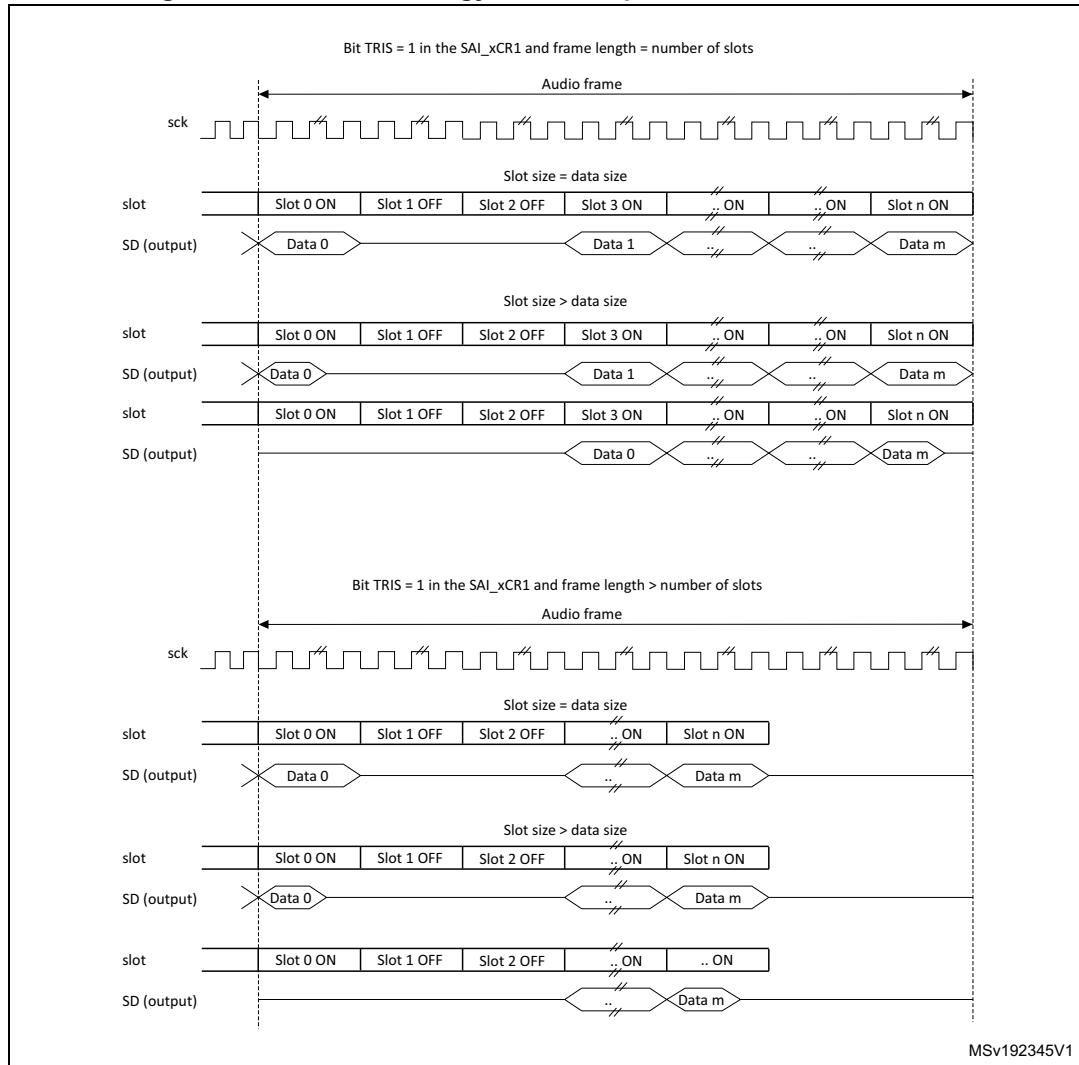
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in HI-z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which may result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting bit SLOTSZ[1:0] = 10 in the SAI\_xSLOTR register. The SD output pin is then tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line is tri-stated when the padding to 0 is done to complete the audio frame.

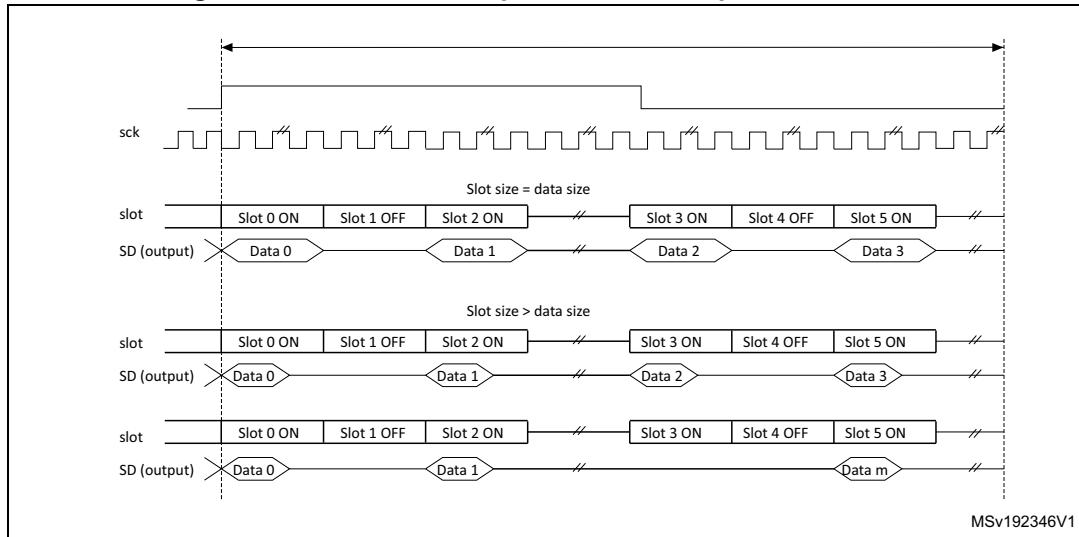
*Figure 394* illustrates these behaviors.

**Figure 394. Tristate strategy on SD output line on an inactive slot**



When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI\_xFRCR register), the tristate mode is managed according to [Figure 395](#) (where bit TRIS in the SAI\_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).

Figure 395. Tristate on output data line in a protocol like I2S



If the TRIS bit in the SAI\_xCR2 register is cleared, all the High impedance states on the SD output line on [Figure 394](#) and [Figure 395](#) are replaced by a drive with a value of 0.

### 36.4.14 Error flags

The SAI implements the following error flags:

- FIFO overrun/underrun
- Anticipated frame synchronization detection
- Late frame synchronization detection
- Codec not ready (AC'97 exclusively)
- Wrong clock configuration in master mode.

#### FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI\_xSR register.

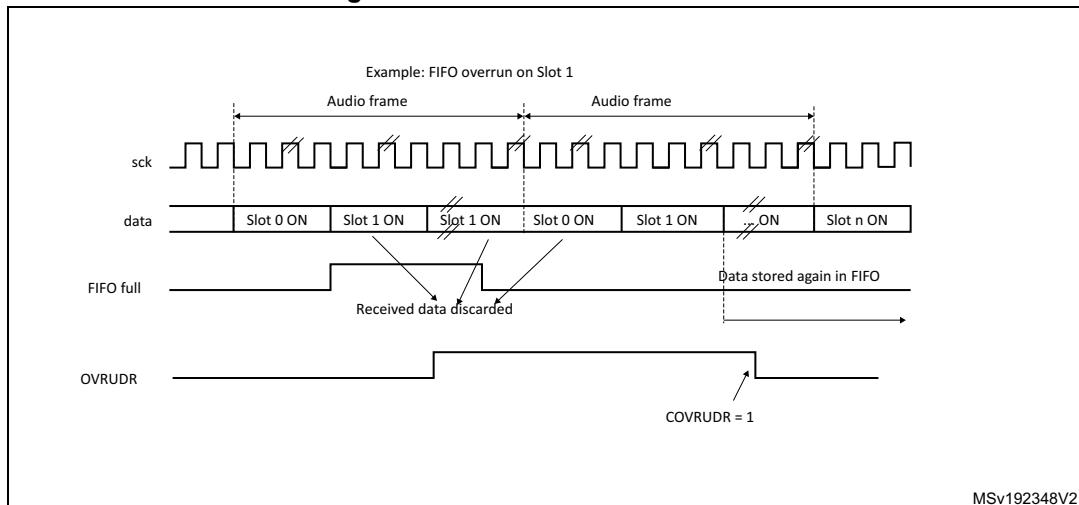
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI\_xSR register.

##### Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the flag OVRUDR in the SAI\_xSR register is set and an interrupt is generated if OVRUDRIE bit is set in the SAI\_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data are stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver stores new data (from new audio frame) from the slot number which was stored internally when the overrun condition was detected. This avoids data slot de-alignment in the destination memory (refer to [Figure 396](#)).

The OVRUDR flag is cleared when COVRUDR bit is set in the SAI\_xCLRFR register.

Figure 396. Overrun detection error



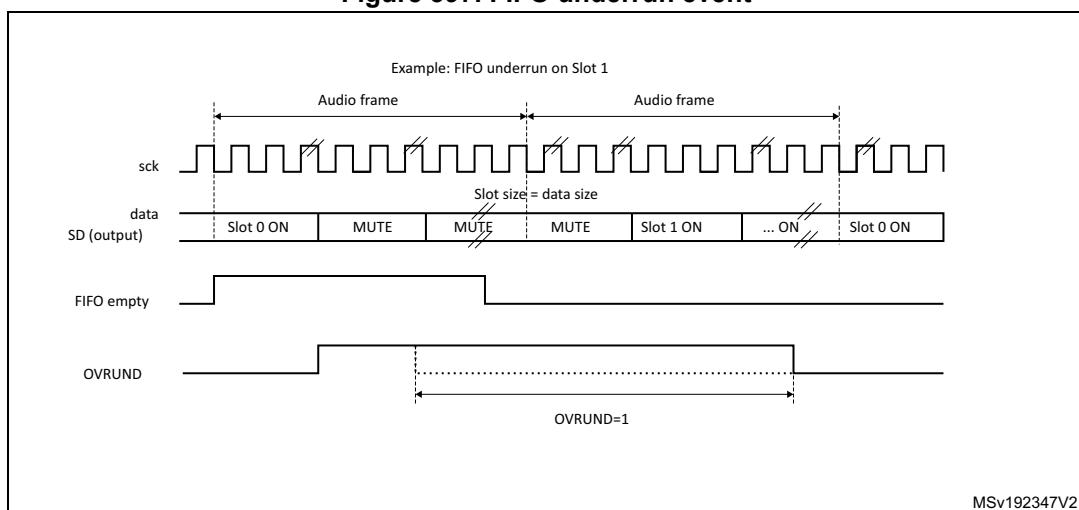
### Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to [Figure 397](#)). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI\_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI\_xIM register. To clear this flag, set COVRUDR bit in the SAI\_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.

Figure 397. FIFO underrun event



### Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI\_xSR register.

This detection has no effect on the current audio frame which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI\_xIM register. To clear the AFSDET flag, CAFSDET bit must be set in the SAI\_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting SAIEN bit in SAI\_xCR1 register. To make sure the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via FFLUS bit in SAI\_xCR2 register.
3. Enable again the SAI peripheral (SAIEN bit set to 1).
4. The SAI block waits for the assertion on FS to restart the synchronization with master.

**Note:** *The AFSDET flag is not asserted in AC'97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave. It has no meaning in SPDIF mode since the FS signal is not used.*

### Late frame synchronization detection

The LFSDET flag in the SAI\_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity and the frame offset configuration are known in register SAI\_xFRCR.

If the external master does not send the FS signal at the expecting time thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if LFSDETIE bit is set in the SAI\_xIM register.

The LFSDET flag is cleared when CLFSDET bit is set in the SAI\_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in [Anticipated frame synchronization detection \(AFSDET\)](#)).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which should not be the case in most applications. In this case, the LFSDET flag is set.

**Note:** *The LFSDET flag is not asserted in AC'97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave. It has no meaning in SPDIF mode since the signal FS is not used by the protocol.*

### Codec not ready (CNRDY AC'97)

The CNRDY flag in the SAI\_xSR register is relevant only if the SAI audio block is configured to operate in AC'97 mode (PRTCFG[1:0] = 10 in the SAI\_xCR1 register). If CNRDYIE bit is set in the SAI\_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the Codec is not ready to communicate during the reception of the TAG 0 (slot0) of the AC'97 audio frame. In this case, no data are automatically stored into the FIFO since the Codec is not ready, until the TAG 0 indicates that the Codec is ready. All the active slots defined in the SAI\_xSLOTR register are captured when the Codec is ready.

To clear CNRDY flag, CCNRDY bit must be set in the SAI\_xCLRFR register.

### Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- (FRL+1) is not a power of 2, and
- (FRL+1) is not between 8 and 256.

MODE, NODIV, and SAIEN bits belong to SAI\_xCR1 register and FRL to SAI\_xFRCR register.

If WCKCFGIE bit is set, an interrupt is generated when WCKCFG flag is set in the SAI\_xSR register. To clear this flag, set CWCKCFG bit in the SAI\_xCLRFR register.

When WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of SAIEN bit.

## 36.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing SAIEN bit in the SAI\_xCR1 register. All the already started frames are automatically completed before the SAI is stops working. SAIEN bit remains High until the SAI is completely switched-off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one which is the master must be disabled first.

## 36.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI\_xDR register (to access the internal FIFO).

There is one DMA channel per audio subblock supporting basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set DMAEN bit in the SAI\_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to [Section 36.4.9: Internal FIFOs](#)). DMA transfer direction is linked to the SAI audio subblock configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI\_xDR register.
- If the audio block is operates as a receiver, the DMA request is related to read operations from the SAI\_xDR register.

Follow the sequence below to configure the SAI interface in DMA mode:

1. Configure SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

*Note:* *Before configuring the SAI block, the SAI DMA channel must be disabled.*

## 36.5 SAI interrupts

The SAI supports 7 interrupt sources as shown in [Table 233](#).

**Table 233. SAI interrupt sources**

Interrupt acronym	Interrupt source	Interrupt group	Audio block mode	Interrupt enable	Interrupt clear
SAI	FREQ	FREQ	Master or slave Receiver or transmitter	FREQIE in SAI_xIM register	Depends on: – FIFO threshold setting (FLVL bits in SAI_xCR2) – Communication direction (transmitter or receiver)  For more details refer to <a href="#">Section 36.4.9: Internal FIFOs</a>
	OVRUDR	ERROR	Master or slave Receiver or transmitter	OVRUDRIE in SAI_xIM register	COVRUDR = 1 in SAI_xCLRFR register
	AFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	AFSDETIE in SAI_xIM register	CAFSDET = 1 in SAI_xCLRFR register
	LFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	LFSDETIE in SAI_xIM register	CLFSDET = 1 in SAI_xCLRFR register
	CNRDY	ERROR	Slave (only in AC'97 mode)	CNRDYIE in SAI_xIM register	CCNRDY = 1 in SAI_xCLRFR register
	MUTEDET	MUTE	Master or slave Receiver mode only	MUTEDETIE in SAI_xIM register	CMUTEDET = 1 in SAI_xCLRFR register
	WCKCFG	ERROR	Master with NODIV = 0 in SAI_xCR1 register	WCKCFGIE in SAI_xIM register	CWCKCFG = 1 in SAI_xCLRFR register

Follow the sequence below to enable an interrupt:

1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.

## 36.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

### 36.6.1 SAI configuration register 1 (SAI\_ACR1)

Address offset: 0x004

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res.	Res.	Res.	MCK EN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res.	OUTDRIV	MONO	SYNCEN[1:0]	CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]		
		rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN**: Master clock generation enable

- 0: The master clock is not generated
- 1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR**: Oversampling ratio for master clock

This bit is meaningful only when NODIV bit is set to 0.

- 0: Master clock frequency =  $F_{FS} \times 256$
- 1: Master clock frequency =  $F_{FS} \times 512$

Bits 25:20 **MCKDIV[5:0]**: Master clock divider

These bits are set and cleared by software.

000000: Divides by 1 the kernel clock input (sai\_x\_ker\_ck).

Otherwise, The master clock frequency is calculated according to the formula given in [Section 36.4.8: SAI clock generator](#).

These bits have no meaning when the audio block is slave.

They have to be configured when the audio block is disabled.

Bit 19 **NODIV**: No divider

This bit is set and cleared by software.

0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512

1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable

This bit is set and cleared by software.

0: DMA disabled

1: DMA enabled

*Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.*

Bit 16 **SAIEN**: Audio block enable

This bit is set by software.

To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.

This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.

0: SAI audio block disabled

1: SAI audio block enabled

*Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.*

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRV**: Output drive

This bit is set and cleared by software.

0: Audio block output driven when SAIEN is set

1: Audio block output driven immediately after the setting of this bit.

*Note: This bit has to be set before enabling the audio block and after the audio block configuration.*

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to [Section : Mono/stereo mode](#) for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]**: Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: Reserved.

11: Reserved

*Note: The audio subblock should be configured as asynchronous when SPDIF mode is enabled.*

Bit 9 **CKSTR**: Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

**Bit 8 LSBFIRST: Least significant bit first**

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

**Bits 7:5 DS[2:0]: Data size**

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFC[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

**Bit 4 Reserved, must be kept at reset value.****Bits 3:2 PRTCFC[1:0]: Protocol configuration**

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

**Bits 1:0 MODE[1:0]: SAIx audio block mode**

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

*Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).*

### 36.6.2 SAI configuration register 1 (SAI\_BCR1)

Address offset: 0x024

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res.	Res.	Res.	MCK EN	OSR			MCKDIV[5:0]				NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res.	OUTD RIV	MONO	SYNCEN[1:0]	CKSTR	LSBFIRST		DS[2:0]		Res.	PRTCFG[1:0]		MODE[1:0]		
		rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN**: Master clock generation enable

0: The master clock is not generated

1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR**: Oversampling ratio for master clock

This bit is meaningful only when NODIV bit is set to 0.

0: Master clock frequency =  $F_{FS} \times 256$

1: Master clock frequency =  $F_{FS} \times 512$

Bits 25:20 **MCKDIV[5:0]**: Master clock divider

These bits are set and cleared by software.

000000: Divides by 1 the kernel clock input (sai\_x\_ker\_ck).

Otherwise, The master clock frequency is calculated according to the formula given in [Section 36.4.8: SAI clock generator](#).

These bits have no meaning when the audio block is slave.

They have to be configured when the audio block is disabled.

Bit 19 **NODIV**: No divider

This bit is set and cleared by software.

0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512

1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable

This bit is set and cleared by software.

0: DMA disabled

1: DMA enabled

*Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.*

Bit 16 **SAIEN**: Audio block enable

This bit is set by software.

To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.

This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.

0: SAI audio block disabled

1: SAI audio block enabled.

*Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.*

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRV**: Output drive

This bit is set and cleared by software.

0: Audio block output driven when SAIEN is set

1: Audio block output driven immediately after the setting of this bit.

*Note: This bit has to be set before enabling the audio block and after the audio block configuration.*

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to [Section : Mono/stereo mode](#) for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]**: Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: Reserved.

11: Reserved

*Note: The audio subblock should be configured as asynchronous when SPDIF mode is enabled.*

Bit 9 **CKSTR**: Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST**: Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]**: Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTC<sub>FG</sub>[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

- 000: Reserved
- 001: Reserved
- 010: 8 bits
- 011: 10 bits
- 100: 16 bits
- 101: 20 bits
- 110: 24 bits
- 111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTC<sub>FG</sub>[1:0]**: Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I<sub>2</sub>S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

- 01: SPDIF protocol
- 10: AC'97 protocol
- 11: Reserved

Bits 1:0 **MODE[1:0]**: SAI<sub>x</sub> audio block mode

These bits are set and cleared by software. They must be configured when SAI<sub>x</sub> audio block is disabled.

- 00: Master transmitter
- 01: Master receiver
- 10: Slave transmitter
- 11: Slave receiver

*Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.*

### 36.6.3 SAI configuration register 2 (SAI\_ACR2)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]	CPL	MUTECNT[5:0]						MUTE <sub>VAL</sub>	MUTE	TRIS	F <sub>FLUSH</sub>	FTH[2:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The  $\mu$ -Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to [Section : Companding mode](#) for more details.

00: No companding algorithm

01: Reserved.

10:  $\mu$ -Law algorithm

11: A-Law algorithm

*Note: Companding mode is applicable only when Free protocol mode is selected.*

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.

1: 2's complement representation.

*Note: This bit has effect only when the companding mode is  $\mu$ -Law algorithm or A-Law algorithm.*

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to [Section : Mute mode](#) for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It must be written before enabling the audio block: SAIEN. This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

If the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to [Section : Mute mode](#) for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

*Note: This bit is meaningless and should not be used for SPDIF audio blocks.*

Bit 5 **MUTE**: Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to [Section : Mute mode](#) for more details.

0: No mute mode.

1: Mute mode enabled.

*Note: This bit is meaningless and should not be used for SPDIF audio blocks.*

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It should be configured when SAI is disabled.

Refer to [Section : Output data line management on an inactive slot](#) for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH**: FIFO flush.

This bit is set by software. It is always read as 0. This bit should be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]**: FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: 1/4 FIFO

010: 1/2 FIFO

011: 3/4 FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

#### 36.6.4 SAI configuration register 2 (SAI\_BCR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]	CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The  $\mu$ -Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to [Section : Companding mode](#) for more details.

00: No companding algorithm

01: Reserved.

10:  $\mu$ -Law algorithm

11: A-Law algorithm

*Note: Companding mode is applicable only when Free protocol mode is selected.*

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.

1: 2's complement representation.

*Note: This bit has effect only when the companding mode is  $\mu$ -Law algorithm or A-Law algorithm.*

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to [Section : Mute mode](#) for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It must be written before enabling the audio block: SAIEN.

This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

If the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to [Section : Mute mode](#) for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

*Note: This bit is meaningless and should not be used for SPDIF audio blocks.*

Bit 5 **MUTE**: Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to [Section : Mute mode](#) for more details.

0: No mute mode.

1: Mute mode enabled.

*Note: This bit is meaningless and should not be used for SPDIF audio blocks.*

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It should be configured when SAI is disabled.

Refer to [Section : Output data line management on an inactive slot](#) for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH**: FIFO flush.

This bit is set by software. It is always read as 0. This bit should be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]**: FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: 1/4 FIFO

010: 1/2 FIFO

011: 3/4 FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

### 36.6.5 SAI frame configuration register (SAI\_AFRCR)

Address offset: 0x00C

Reset value: 0x0000 0007

*Note:* This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.

This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.

0: FS is asserted on the first bit of the slot 0.

1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.

This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.

This bit must be configured when the audio block is disabled.

0: FS is active low (falling edge)

1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.

This bit is set and cleared by software.

0: FS signal is a start frame signal

1: FS signal is a start of frame signal + channel side identification

When the bit is set, the number of slots defined in the SAI\_xSLOTR register has to be even. It means that half of this number of slots are dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).

This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.

These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI\_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK\_x pin) is used, the frame length should be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.

### 36.6.6 SAI frame configuration register (SAI\_BFRCR)

Address offset: 0x02C

Reset value: 0x0000 0007

*Note:* This register has no meaning in AC'97 and SPDIF audio protocol

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.

This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.

0: FS is asserted on the first bit of the slot 0.

1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.

This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.

This bit must be configured when the audio block is disabled.

0: FS is active low (falling edge)

1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.

This bit is set and cleared by software.

0: FS signal is a start frame signal

1: FS signal is a start of frame signal + channel side identification

When the bit is set, the number of slots defined in the SAI\_xSLOTR register has to be even. It means that half of this number of slots is dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).

This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.

These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI\_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK\_x pin) is used, the frame length should be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256. These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

### 36.6.7 SAI slot register (SAI\_ASLOTR)

Address offset: 0x010

Reset value: 0x0000 0000

**Note:** *This register has no meaning in AC'97 and SPDIF audio protocol.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.

These bits are set and cleared by software.

Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).

0: Inactive slot.

1: Active slot.

The slot must be enabled when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.

These bits are set and cleared by software.

The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.

The number of slots should be even if FSDEF bit in the SAI\_xFRCR register is set.

The number of slots must be configured when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

Bits 7:6 **SLOTSZ[1:0]**: Slot size

This bits is set and cleared by software.

The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.

Refer to [Output data line management on an inactive slot](#) for information on how to drive SD line.

These bits must be set when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI\_xCR1 register).

01: 16-bit

10: 32-bit

11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset

These bits are set and cleared by software.

The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.

These bits must be set when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

**36.6.8 SAI slot register (SAI\_BSLOTR)**

Address offset: 0x030

Reset value: 0x0000 0000

*Note:* *This register has no meaning in AC'97 and SPDIF audio protocol.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.

These bits are set and cleared by software.

Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).

0: Inactive slot.

1: Active slot.

The slot must be enabled when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.

These bits are set and cleared by software.

The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.

The number of slots should be even if FSDEF bit in the SAI\_xFRCR register is set.

The number of slots must be configured when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

Bits 7:6 **SLOTSZ[1:0]**: Slot size

These bits are set and cleared by software.

The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.

Refer to [Output data line management on an inactive slot](#) for information on how to drive SD line.

These bits must be set when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI\_xCR1 register).

01: 16-bit

10: 32-bit

11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset

These bits are set and cleared by software.

The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.

These bits must be set when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

### 36.6.9 SAI interrupt mask register (SAI\_AIM)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LFSDET IE	AFSDETI E	CNRDY IE	FREQ IE	WCKCFG IE	MUTEDET IE	OVRUDR IE								
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI\_xSR register.

This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the AFSDET bit in the SAI\_xSR register is set.

This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI\_xSR register is set and an interrupt is generated.

This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the FREQ bit in the SAI\_xSR register is set.

Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode.

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.

It generates an interrupt if the WCKCFG flag in the SAI\_xSR register is set.

*Note: This bit is used only in Free protocol mode and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI\_xSR register is set.

This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI\_xSR register is set.

### 36.6.10 SAI interrupt mask register (SAI\_BIM)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LFSDET IE	AFSDETI E	CNRDY IE	FREQ IE	WCKCFG IE	MUTEDET IE	OVRUDR IE								
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI\_xSR register.

This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the AFSDET bit in the SAI\_xSR register is set.

This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI\_xSR register is set and an interrupt is generated.

This bit has a meaning only if the AC'97 mode is selected through PRTCFCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the FREQ bit in the SAI\_xSR register is set.

Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.

It generates an interrupt if the WCKCFG flag in the SAI\_xSR register is set.

*Note: This bit is used only in Free protocol mode and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI\_xSR register is set.

This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI\_xSR register is set.

### 36.6.11 SAI status register (SAI\_ASR)

Address offset: 0x018

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	Res.	FLVL[2:0]											
														r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR									
									r	r	r	r	r	r	r	

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.

This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).

000: FIFO empty (transmitter and receiver modes)

001: FIFO  $\leq \frac{1}{4}$  but not empty (transmitter mode), FIFO  $< \frac{1}{4}$  but not empty (receiver mode)

010:  $\frac{1}{4} < \text{FIFO} \leq \frac{1}{2}$  (transmitter mode),  $\frac{1}{4} \leq \text{FIFO} < \frac{1}{2}$  (receiver mode)

011:  $\frac{1}{2} < \text{FIFO} \leq \frac{3}{4}$  (transmitter mode),  $\frac{1}{2} \leq \text{FIFO} < \frac{3}{4}$  (receiver mode)

100:  $\frac{3}{4} < \text{FIFO}$  but not full (transmitter mode),  $\frac{3}{4} \leq \text{FIFO}$  but not full (receiver mode)

101: FIFO full (transmitter and receiver modes)

Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.

This bit is read only.

0: No error.

1: Frame synchronization signal is not present at the right time.

This flag can be set only if the audio block is configured in slave mode.

It is not used in AC'97 or SPDIF mode.

It can generate an interrupt if LFSDETIE bit is set in the SAI\_xIM register.

This flag is cleared when the software sets bit CLFSDET in SAI\_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.

This bit is read only.

0: No error.

1: Frame synchronization signal is detected earlier than expected.

This flag can be set only if the audio block is configured in slave mode.

It is not used in AC'97 or SPDIF mode.

It can generate an interrupt if AFSDETIE bit is set in SAI\_xIM register.

This flag is cleared when the software sets CAFSDET bit in SAI\_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.

This bit is read only.

0: External AC'97 Codec is ready

1: External AC'97 Codec is not ready

This bit is used only when the AC'97 audio protocol is selected in the SAI\_xCR1 register and configured in receiver mode.

It can generate an interrupt if CNRDYIE bit is set in SAI\_xIM register.

This flag is cleared when the software sets CCNRDY bit in SAI\_xCLRFR register.

Bit 3 **FREQ**: FIFO request.

This bit is read only.

0: No FIFO request.

1: FIFO request to read or to write the SAI\_xDR.

The request depends on the audio block configuration:

- If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI\_xDR.
- If the block configured in reception, the FIFO request related to a read request operation from the SAI\_xDR.

This flag can generate an interrupt if FREQIE bit is set in SAI\_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.

This bit is read only.

0: Clock configuration is correct

1: Clock configuration does not respect the rule concerning the frame length specification defined in [Section 36.4.6: Frame synchronization](#) (configuration of FRL[7:0] bit in the SAI\_xFRCR register)

This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.

It can generate an interrupt if WCKCFGIE bit is set in SAI\_xIM register.

This flag is cleared when the software sets CWCKCFG bit in SAI\_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.

This bit is read only.

0: No MUTE detection on the SD input line

1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame

This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI\_xCR2 register).

It can generate an interrupt if MUTEDETIE bit is set in SAI\_xIM register.

This flag is cleared when the software sets bit CMUTEDET in the SAI\_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.

This bit is read only.

0: No overrun/underrun error.

1: Overrun/underrun error detection.

The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.

It can generate an interrupt if OVRUDRIE bit is set in SAI\_xIM register.

This flag is cleared when the software sets COVRUDR bit in SAI\_xCLRFR register.

### 36.6.12 SAI status register (SAI\_BSR)

Address offset: 0x038

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	FLVL[2:0]		
Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR											

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.

This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).

000: FIFO empty (transmitter and receiver modes)

001: FIFO  $\leq \frac{1}{4}$  but not empty (transmitter mode), FIFO  $< \frac{1}{4}$  but not empty (receiver mode)

010:  $\frac{1}{4} < \text{FIFO} \leq \frac{1}{2}$  (transmitter mode),  $\frac{1}{4} \leq \text{FIFO} < \frac{1}{2}$  (receiver mode)

011:  $\frac{1}{2} < \text{FIFO} \leq \frac{3}{4}$  (transmitter mode),  $\frac{1}{2} \leq \text{FIFO} < \frac{3}{4}$  (receiver mode)

100:  $\frac{3}{4} < \text{FIFO}$  but not full (transmitter mode),  $\frac{3}{4} \leq \text{FIFO}$  but not full (receiver mode)

101: FIFO full (transmitter and receiver modes)

Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.

This bit is read only.

0: No error.

1: Frame synchronization signal is not present at the right time.

This flag can be set only if the audio block is configured in slave mode.

It is not used in AC'97 or SPDIF mode.

It can generate an interrupt if LFSDETIE bit is set in the SAI\_xIM register.

This flag is cleared when the software sets bit CLFSDET in SAI\_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.

This bit is read only.

0: No error.

1: Frame synchronization signal is detected earlier than expected.

This flag can be set only if the audio block is configured in slave mode.

It is not used in AC'97 or SPDIF mode.

It can generate an interrupt if AFSDETIE bit is set in SAI\_xIM register.

This flag is cleared when the software sets CAFSDET bit in SAI\_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.

This bit is read only.

0: External AC'97 Codec is ready

1: External AC'97 Codec is not ready

This bit is used only when the AC'97 audio protocol is selected in the SAI\_xCR1 register and configured in receiver mode.

It can generate an interrupt if CNRDYIE bit is set in SAI\_xIM register.

This flag is cleared when the software sets CCNRDY bit in SAI\_xCLRFR register.

Bit 3 **FREQ**: FIFO request.

This bit is read only.

0: No FIFO request.

1: FIFO request to read or to write the SAI\_xDR.

The request depends on the audio block configuration:

- If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI\_xDR.
- If the block configured in reception, the FIFO request related to a read request operation from the SAI\_xDR.

This flag can generate an interrupt if FREQIE bit is set in SAI\_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.

This bit is read only.

0: Clock configuration is correct

1: Clock configuration does not respect the rule concerning the frame length specification defined in [Section 36.4.6: Frame synchronization](#) (configuration of FRL[7:0] bit in the SAI\_xFRCR register)

This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.

It can generate an interrupt if WCKCFGIE bit is set in SAI\_xIM register.

This flag is cleared when the software sets CWCKCFG bit in SAI\_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.

This bit is read only.

0: No MUTE detection on the SD input line

1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame

This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI\_xCR2 register).

It can generate an interrupt if MUTEDETIE bit is set in SAI\_xIM register.

This flag is cleared when the software sets bit CMUTEDET in the SAI\_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.

This bit is read only.

0: No overrun/underrun error.

1: Overrun/underrun error detection.

The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.

It can generate an interrupt if OVRUDRIE bit is set in SAI\_xIM register.

This flag is cleared when the software sets COVRUDR bit in SAI\_xCLRFR register.

### 36.6.13 SAI clear flag register (SAI\_ACLRFR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR								
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.

This bit is write only.

Programming this bit to 1 clears the LFSDET flag in the SAI\_xSR register.

This bit is not used in AC'97 or SPDIF mode

Reading this bit always returns the value 0.

Bit 5 **CAFSDDET**: Clear anticipated frame synchronization detection flag.

This bit is write only.

Programming this bit to 1 clears the AFSDET flag in the SAI\_xSR register.

It is not used in AC'97 or SPDIF mode.

Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.

This bit is write only.

Programming this bit to 1 clears the CNRDY flag in the SAI\_xSR register.

This bit is used only when the AC'97 audio protocol is selected in the SAI\_xCR1 register.

Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.

This bit is write only.

Programming this bit to 1 clears the WCKCFG flag in the SAI\_xSR register.

This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI\_xCR1 register.

Reading this bit always returns the value 0.

Bit 1 **CMUTEDDET**: Mute detection flag.

This bit is write only.

Programming this bit to 1 clears the MUTEDDET flag in the SAI\_xSR register.

Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.

This bit is write only.

Programming this bit to 1 clears the OVRUDR flag in the SAI\_xSR register.

Reading this bit always returns the value 0.

### 36.6.14 SAI clear flag register (SAI\_BCLRFR)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CLFSDET	CAFSDDET	CCNRDY	Res.	CWCKCFG	CMUTEDDET	COVRUDR								
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.

This bit is write only.

Programming this bit to 1 clears the LFSDET flag in the SAI\_xSR register.

This bit is not used in AC'97 or SPDIF mode

Reading this bit always returns the value 0.

Bit 5 **CAFSDDET**: Clear anticipated frame synchronization detection flag.

This bit is write only.

Programming this bit to 1 clears the AFSDET flag in the SAI\_xSR register.

It is not used in AC'97 or SPDIF mode.

Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.

This bit is write only.

Programming this bit to 1 clears the CNRDY flag in the SAI\_xSR register.

This bit is used only when the AC'97 audio protocol is selected in the SAI\_xCR1 register.

Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.

This bit is write only.

Programming this bit to 1 clears the WCKCFG flag in the SAI\_xSR register.

This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI\_xCR1 register.

Reading this bit always returns the value 0.

Bit 1 **CMUTEDDET**: Mute detection flag.

This bit is write only.

Programming this bit to 1 clears the MUTEDDET flag in the SAI\_xSR register.

Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.

This bit is write only.

Programming this bit to 1 clears the OVRUDR flag in the SAI\_xSR register.

Reading this bit always returns the value 0.

### 36.6.15 SAI data register (SAI\_ADR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[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
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

A write to this register loads the FIFO provided the FIFO is not full.

A read from this register empties the FIFO if the FIFO is not empty.

### 36.6.16 SAI data register (SAI\_BDR)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[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
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

A write to this register loads the FIFO provided the FIFO is not full.

A read from this register empties the FIFO if the FIFO is not empty.

### 36.6.17 SAI PDM control register (SAI\_PDMCR)

Address offset: 0x0044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]	Res.	Res.	Res.	PDMEN	
						rw	rw			rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2

This bit is set and cleared by software.

0: SAI\_CK2 clock disabled

1: SAI\_CK2 clock enabled

*Note: It is not recommended to configure this bit when PDMEN = 1.*

*SAI\_CK2 might not be available for all SAI instances. Refer to [Section 36.3: SAI implementation](#) for details.*

Bit 8 **CKEN1**: Clock enable of bitstream clock number 1

This bit is set and cleared by software.

0: SAI\_CK1 clock disabled

1: SAI\_CK1 clock enabled

*Note: It is not recommended to configure this bit when PDMEN = 1.*

*SAI\_CK1 might not be available for all SAI instances. Refer to [Section 36.3: SAI implementation](#) for details.*

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MICNBR[1:0]**: Number of microphones

This bit is set and cleared by software.

00: Configuration with 2 microphones

01: Configuration with 4 microphones

10: Configuration with 6 microphones

11: Configuration with 8 microphones

*Note: It is not recommended to configure this field when PDMEN = 1.\**

*The complete set of data lines might not be available for all SAI instances. Refer to [Section 36.3: SAI implementation](#) for details.*

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable

This bit is set and cleared by software. This bit enables to control the state of the PDM interface block.

Make sure that the SAI is already operating in TDM master mode before enabling the PDM interface.

0: PDM interface disabled

1: PDM interface enabled

### 36.6.18 SAI PDM delay register (SAI\_PDMDLY)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]		
	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	rw	rw	rw												

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**

This bit is set and cleared by software.

000: No delay

001: Delay of 1  $T_{SAI\_CK}$  period

010: Delay of 2  $T_{SAI\_CK}$  periods

...

111: Delay of 7  $T_{SAI\_CK}$  periods

This field can be changed on-the-fly.

*Note: This field can be used only if D4 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **DLYM4L[2:0]**: Delay line for first microphone of pair 4

This bit is set and cleared by software.

000: No delay

001: Delay of 1  $T_{SAI\_CK}$  period

010: Delay of 2  $T_{SAI\_CK}$  periods

...

111: Delay of 7  $T_{SAI\_CK}$  periods

This field can be changed on-the-fly.

*Note: This field can be used only if D4 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DLYM3R[2:0]**: Delay line for second microphone of pair 3

This bit is set and cleared by software.

000: No delay

001: Delay of 1  $T_{SAI\_CK}$  period

010: Delay of 2  $T_{SAI\_CK}$  periods

...

111: Delay of 7  $T_{SAI\_CK}$  periods

This field can be changed on-the-fly.

*Note: This field can be used only if D3 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DLYM3L[2:0]**: Delay line for first microphone of pair 3

This bit is set and cleared by software.

000: No delay

001: Delay of 1  $T_{SAI\_CK}$  period

010: Delay of 2  $T_{SAI\_CK}$  periods

...

111: Delay of 7  $T_{SAI\_CK}$  periods

This field can be changed on-the-fly.

*Note: This field can be used only if D3 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2

This bit is set and cleared by software.

000: No delay

001: Delay of 1  $T_{SAI\_CK}$  period

010: Delay of 2  $T_{SAI\_CK}$  periods

...

111: Delay of 7  $T_{SAI\_CK}$  periods

This field can be changed on-the-fly.

*Note: This field can be used only if D2 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2

This bit is set and cleared by software.

000: No delay

001: Delay of 1  $T_{SAI\_CK}$  period

010: Delay of 2  $T_{SAI\_CK}$  periods

...

111: Delay of 7  $T_{SAI\_CK}$  periods

This field can be changed on-the-fly.

*Note: This field can be used only if D2 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1

This bit is set and cleared by software.

000: No delay

001: Delay of 1  $T_{SAI\_CK}$  period

010: Delay of 2  $T_{SAI\_CK}$  periods

...

111: Delay of 7  $T_{SAI\_CK}$  periods

This field can be changed on-the-fly.

*Note: This field can be used only if D1 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1

This bit is set and cleared by software.

000: No delay

001: Delay of 1  $T_{SAI\_CK}$  period

010: Delay of 2  $T_{SAI\_CK}$  periods

...

111: Delay of 7  $T_{SAI\_CK}$  periods

This field can be changed on-the-fly.

*Note: This field can be used only if D1 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

### 36.6.19 SAI register map

Table 234. SAI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x0000																																		
0x0004 or 0x0024	SAI_xCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0008 or 0x0028	SAI_xCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x000C or 0x002C	SAI_xFRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0010 or 0x0030	SAI_xSLOTR	SLOTEN[15:0]															FSALL[6:0]				FRL[7:0]				CKSTR[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		
0x0014 or 0x0034	SAI_xIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0018 or 0x0038	SAI_xSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x001C or 0x003C	SAI_xCLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0020 or 0x0040	SAI_xDR	DATA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0044	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 234. SAI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0048	SAI_PDMDLY	Res.		DLYM4R[2:0]		Res.		DLYM4L[2:0]		Res.		DLYM3R[2:0]		Res.		DLYM3L[2:0]		Res.		DLYM2R[2:0]		Res.		DLYM2L[2:0]		Res.		DLYM1R[2:0]		Res.		DLYM1L[2:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to [Section 2.2](#) for the register boundary addresses.

## 37 Inter-processor communication controller (IPCC)

### 37.1 IPCC introduction

The inter-processor communication controller (IPCC) is used for communicating data between two processors.

The IPCC block provides a non blocking signaling mechanism to post and retrieve communication data in an atomic way. It provides the signaling for twelve channels:

- six channels in the direction from processor 1 to processor 2
- six channels in the opposite direction

It is then possible to have two different communication types in each direction.

The IPCC communication data must be located in a common memory, which is not part of the IPCC block.

### 37.2 IPCC main features

- Status signaling for the twelve channels
  - Channel occupied/free flag, also used as lock
- Two interrupt lines per processor
  - One for RX channel occupied (communication data posted by sending processor)
  - One for TX channel free (communication data retrieved by receiving processor)
- Interrupt masking per channel
  - Channel occupied mask
  - Channel free mask
- Two channel operation modes
  - Simplex (each channel has its own communication data memory location)
  - Half duplex (a single channel associated to a bidirectional communication data information memory location)

### 37.3 IPCC functional description

The IPCC communication data is located in a common memory, which is not part of the IPCC block. The address location of the communication data must be known or located in a known common area that, as already stated, is not part of the IPCC block.

For each communication, the IPCC block provides a channel status flag CHnF.

- When 0, the channel status flag CHnF indicates that the associated IPCC channel is free (communication data has been retrieved by the receiving processor), and can be accessed by the sending processor.
- When 1, the channel status flag CHnF indicates that the associated IPCC channel is occupied (communication data has been posted by the sending processor) and can be accessed by the receiving processor.

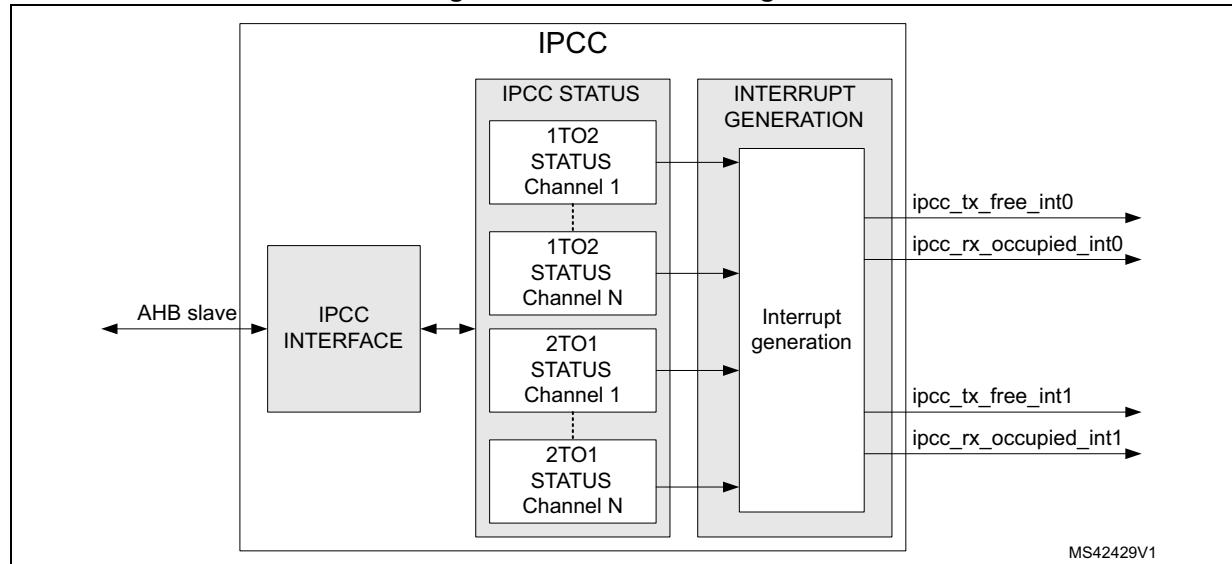
The channel operation mode must be known to both processors. A common parameter can be used to indicate the channel transfer mode and must also be located in a known common area. This parameter is not available from the IPCC.

### 37.3.1 IPCC block diagram

The IPCC (see [Figure 398](#)) consists of the following sub-blocks:

- Status block, containing the channel status
- IPCC interface block, providing AHB access to the channel status registers
- Interrupt interface block, providing control for the interrupts

**Figure 398. IPCC block diagram**



**Table 235. IPCC interface signals**

Signal		Description
Name	Type	
AHB slave	I/O	AHB register access bus
ipcc_tx_free_int1	O	TX free interrupt to processor 1
ipcc_rx_occupied_int1	O	RX occupied interrupt to processor 1
ipcc_tx_free_int2	O	TX free interrupt to processor 2
ipcc_rx_occupied_int2	O	RX occupied interrupt to processor 2

### 37.3.2 IPCC Simplex channel mode

In Simplex channel mode, a dedicated memory location (used to transfer data in a single direction) is assigned to the communication data. The associated channel N control bits (see [Table 236](#)) are used to manage the transfer from the sending to the receiving processor.

**Table 236. Bits used for the communication**

Processor	A	B
SEND A = 1 RECEIVE B = 2	IPCC_C1CR.TXFIE IPCC_C1MR.CHnFM IPCC_C1SCR.CHnS IPCC_C1TOC2SR.CHnF	IPCC_C2CR.RXOIE IPCC_C2MR.CHnOM IPCC_C2SCR.CHnC
SEND A = 2 RECEIVE B = 1	IPCC_C2CR.TXFIE IPCC_C2MR.CHnFM IPCC_C2SCR.CHnS IPCC_C2TOC1SR.CHnF	IPCC_C1CR.RXOIE IPCC_C1MR.CHnOM IPCC_C1SCR.CHnC

Once the sending processor has posted the communication data in the memory, it sets the channel status flag CHnF to occupied with CHnS.

Once the receiving processor has retrieved the communication data from the memory, it clears the channel status flag CHnF back to free with CHnC.

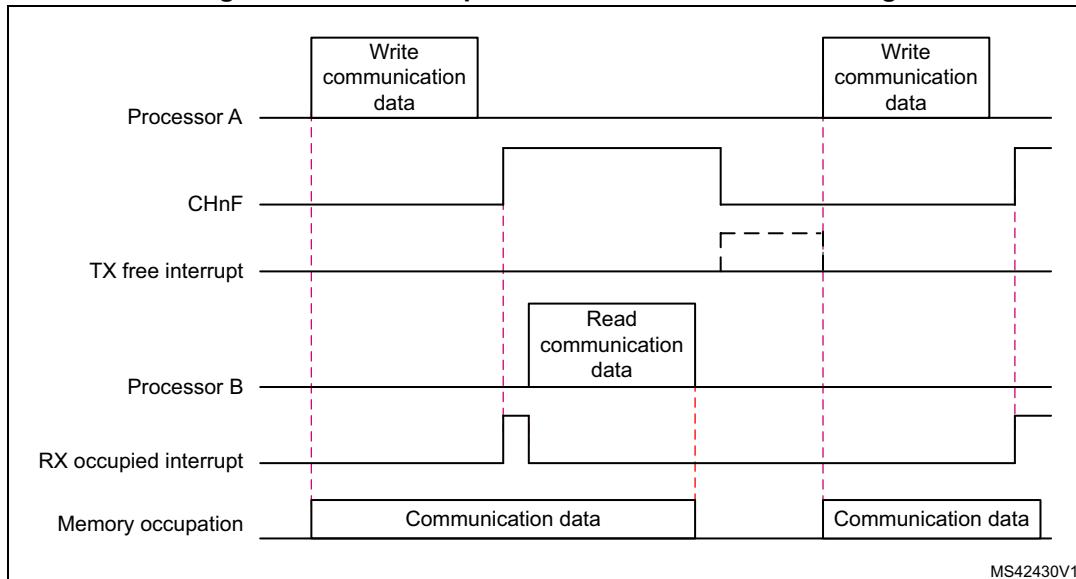
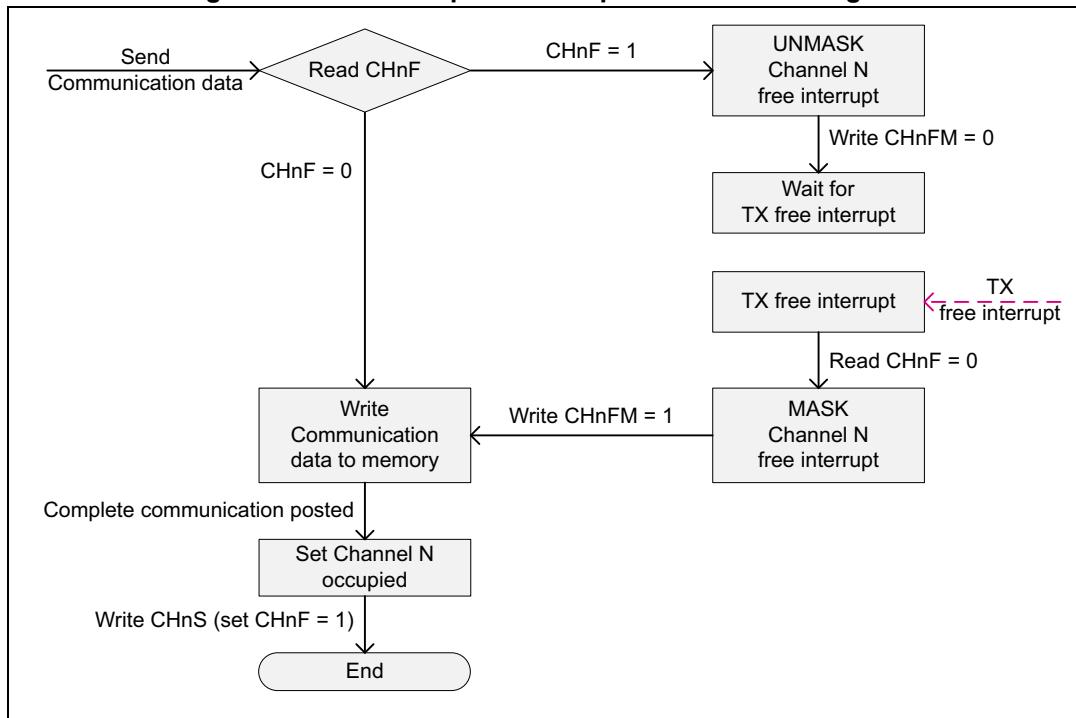
**Figure 399. IPCC Simplex channel mode transfer timing**

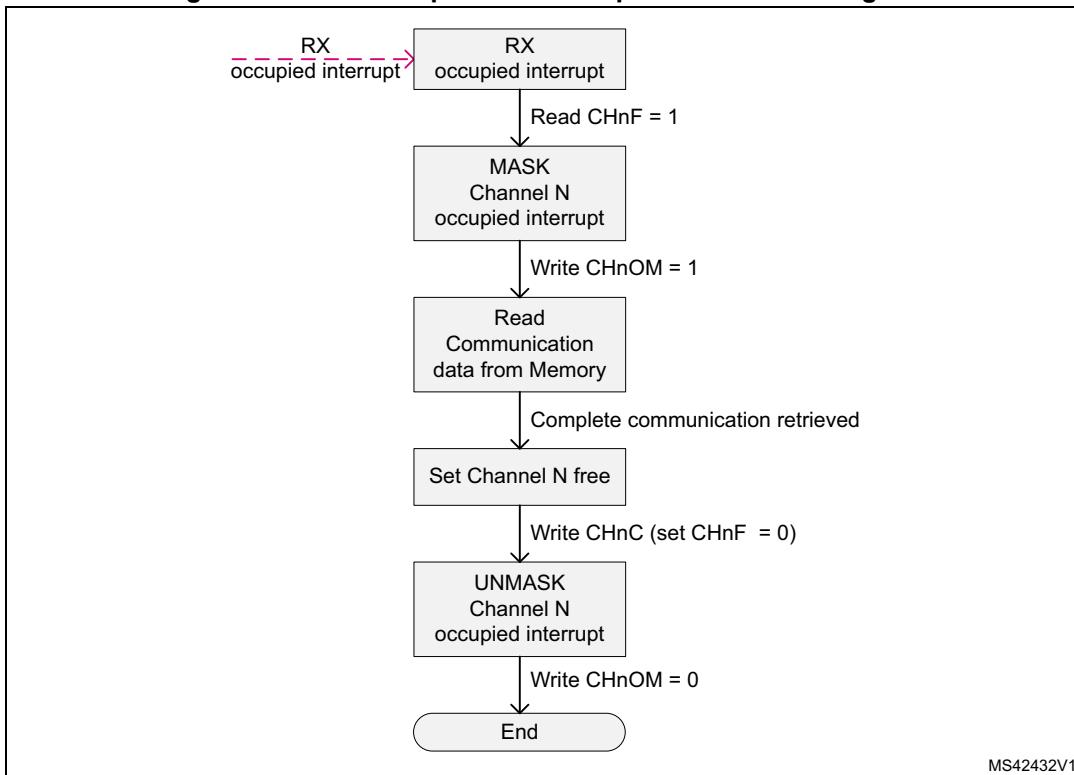
Figure 400. IPCC Simplex - Send procedure state diagram



To send communication data:

- The sending processor checks the channel status flag CHnF:
  - When CHnF = 0, the channel is free (last communication data retrieved by receiving processor) and the new communication data can be written.
  - When CHnF = 1, the channel is occupied (last communication data not retrieved by receiving processor) and the sending processor unmasks the channel free interrupt (CHnFM = 0).
  - On a TX free interrupt, the sending processor checks which channel became free and masks the channel free interrupt (CHnFM = 1). Then the new communication can take place.
- Once the complete communication data is posted, the channel status is set to occupied with CHnS. This gives memory access to the receiving processor and generates the RX occupied interrupt.

Figure 401. IPCC Simplex - Receive procedure state diagram



To receive a communication, the channel occupied interrupt is unmasked (CHnOM = 0):

- On a RX occupied interrupt, the receiving processor checks which channel became occupied, masks the associated channel occupied interrupt (CHnOM) and reads the communication data from memory.
- Once the complete communication data is retrieved, the channel status is cleared to free with CHnC. This gives memory access back to the sending processor and may generate the TX free interrupt.
- Once the channel status is cleared, the channel occupied interrupt is unmasked (CHnOM = 0).

### 37.3.3 IPCC Half-duplex channel mode

The Half-duplex channel mode is used when one processor sends a communication and the other processor sends a response to each communication (ping-pong).

In Half-duplex channel mode, a single dedicated memory location is assigned to communication data and response, and is used to transfer data in both directions. The sending processor channel status flag CHnF is assigned to the channel and used by both processors (see [Table 236](#)).

Once the processor A posted the communication data into memory, it sets the processor A channel status flag CHnF to occupied with CHnS (giving memory access to processor B).

Once the processor B retrieved the communication data from memory, it does not change the channel status flags. The memory access is kept by processor B for the response.

Once the processor B posted the response into memory, it clears the channel status flag CHnF to free with CHnC (giving memory access back to processor A).

Once the processor A retrieved the response from the memory, it does not change the channel status flags. The memory location access is kept by processor A for the next communication data.

Figure 402. IPCC Half-duplex channel mode transfer timing

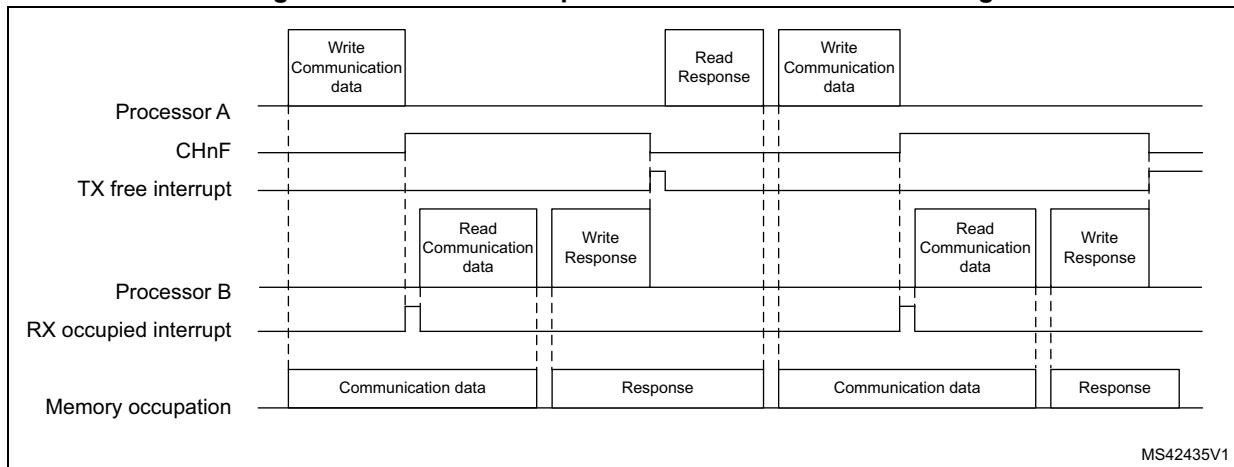
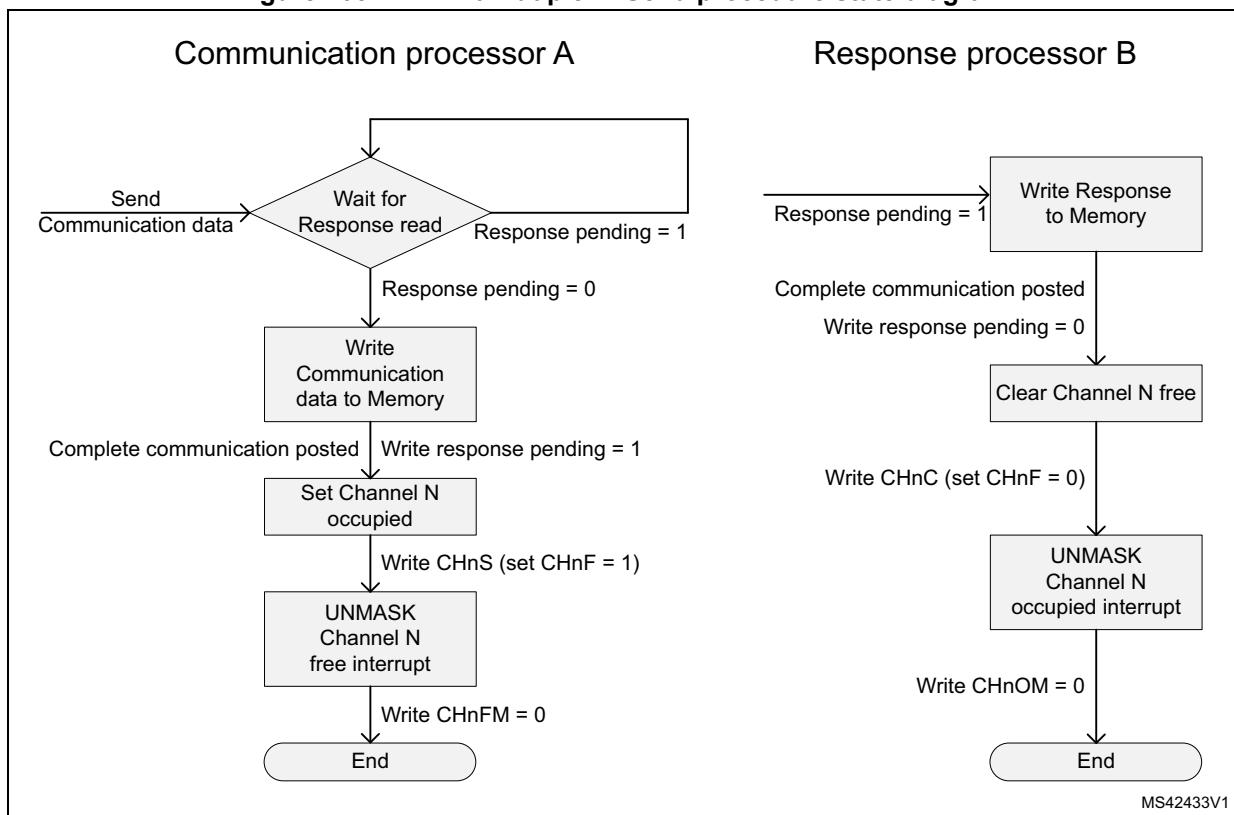


Figure 403. IPCC Half-duplex - Send procedure state diagram



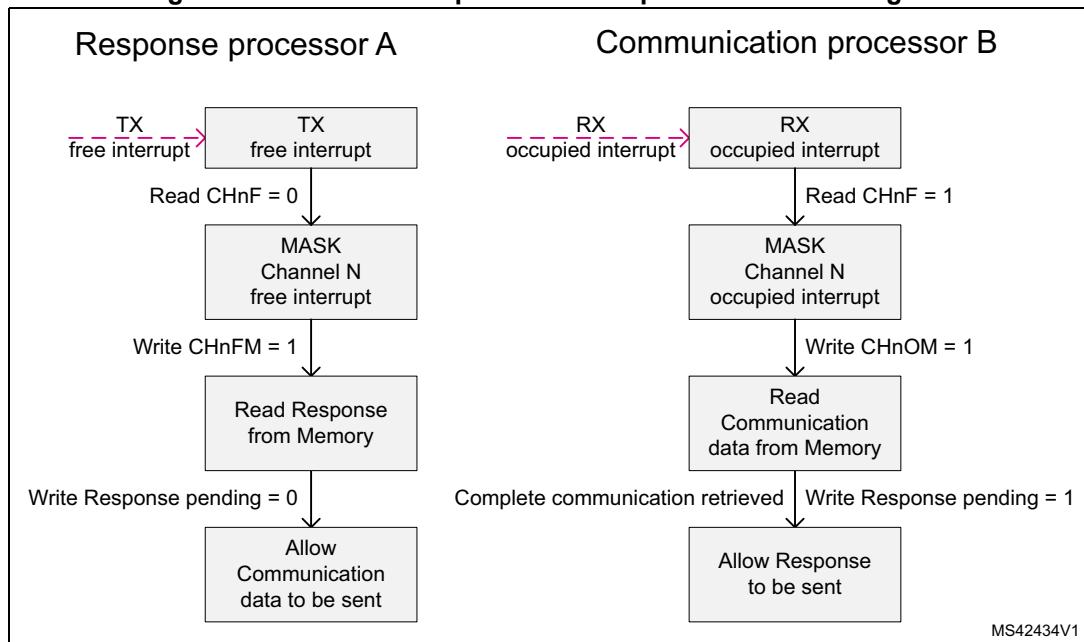
To send communication data:

- The sending processor waits for its response pending software variable to get 0.
  - Once the response pending software variable is 0 the communication data is posted.
- Once the complete communication data has been posted, the channel status flag CHnF is set to occupied with CHnS and the response pending software variable is set to 1 (this gives memory access and generates the RX occupied interrupt to the receiving processor).
- Once the channel status flag CHnF is set, the channel free interrupt is unmasked (CHnFM = 0).

To send a response:

- The receiving processor waits for its response pending software variable to get 1.
  - Once the response pending software variable is 1 the response is posted.
- Once the complete response is posted, the channel status flag CHnF is cleared to free with CHnC and the response pending software variable is set to 0 (this gives memory access and generates the TX free interrupt to the sending processor).
- Once the channel status flag CHnF is cleared, the channel occupied interrupt is unmasked (CHnOM = 0).

**Figure 404. IPCC Half-duplex - Receive procedure state diagram**



To receive communication data the channel occupied interrupt is unmasked (CHnOM = 0):

- On a RX occupied interrupt, the receiving processor checks which channel became occupied, masks the associated channel occupied interrupt (CHnOM) and reads the communication data from the memory.
- Once the complete communication data is retrieved, the response pending software variable is set. The channel status is not changed, access to the memory is kept to post the subsequent response.

To receive the response the channel free interrupt is unmasked (CHnFM = 0):

- On a TX free interrupt, the sending processor checks which channel became free, masks the associated channel free interrupt (CHnFM) and reads the response from the memory.
- Once the complete response is retrieved, the response pending software variable is cleared. The channel status is not changed, access to the memory is kept to post the subsequent communication data.

### 37.3.4 IPCC interrupts

There are four interrupt lines :

- two RX channel occupied interrupts, one for each processor:
  - Interrupt enable RXOIE per processor
  - Individual mask CHnOM per channel
- two TX channel free interrupts, one for each processor
  - Interrupt enable TXFIE per processor
  - Individual mask CHnFM per channel

The RX occupied interrupt is used by the receiving processor and indicates when an unmasked channel status indicates occupied (CHnF = 1).

The TX free interrupt is used by the sending processor, and indicates when an unmasked channel status indicates free (CHnF = 0).

A secure channel only generates a secure interrupt, and only in the case when the channel is secure unmasked and global secure enabled.

A non-secure channel only generates a non-secure interrupt, and only in the case when the channel is non-secure unmasked and global non-secure enabled.

## 37.4 IPCC registers

The peripheral registers must be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses are not permitted and do not generate a bus error.

### 37.4.1 IPCC processor 1 control register (IPCC\_C1CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	TXFIE															
															rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	RXOIE															
															rw	

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **TXFIE**: Processor 1 transmit channel free interrupt enable

Associated with IPCC\_C1TOC2SR.

1: Enable an unmasked processor 1 transmit channel free to generate a TX free interrupt.

0: Processor 1 TX free interrupt disabled

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **RXOIE**: Processor 1 receive channel occupied interrupt enable

Associated with IPCC\_C2TOC1SR.

1: Enable an unmasked processor 1 receive channel occupied to generate an RX occupied interrupt.

0: Processor 1 RX occupied interrupt disabled

### 37.4.2 IPCC processor 1 mask register (IPCC\_C1MR)

Address offset: 0x004

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CH6 FM	CH5 FM	CH4 FM	CH3 FM	CH2 FM	CH1 FM									
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CH6 OM	CH5 OM	CH4 OM	CH3 OM	CH2 OM	CH1 OM									
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnFM**: Processor 1 transmit channel n status set, (n = 6 to 1).

Associated with IPCC\_C1TOC2SR.CHnF

1: Transmit channel n free interrupt masked.

0: Transmit channel n free interrupt not masked.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnOM**: Processor 1 receive channel n status clear (n = 6 to 1).

Associated with IPCC\_C2TOC1SR.CHnF

1: Receive channel n occupied interrupt masked.

0: Receive channel n occupied interrupt not masked.

### 37.4.3 IPCC processor 1 status set clear register (IPCC\_C1SCR)

Address offset: 0x008

Reset value: 0x0000 0000

Reading this register always returns 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CH6S	CH5S	CH4S	CH3S	CH2S	CH1S									
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CH6C	CH5C	CH4C	CH3C	CH2C	CH1C									
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnS**: Processor 1 transmit channel n status set (n = 6 to 1).

Associated with IPCC\_C1TOC2SR.CHnF

1: Processor 1 transmit channel n status bit set.

0: No action.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnC**: Processor 1 receive channel n status clear (n = 6 to 1).

Associated with IPCC\_C2TOC1SR.CHnF

1: Processor 1 receive channel n status bit clear.

0: No action.

### 37.4.4 IPCC processor 1 to processor 2 status register (IPCC\_C1TOC2SR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F									
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnF**: Processor 1 transmit to processor 2 receive channel n status flag before masking (n = 6 to 1).

1: Channel occupied, data can be read by the receiving processor 2.

Generates a channel RX occupied interrupt to processor 2, when unmasked.

0: Channel free, data can be written by the sending processor 1.

Generates a channel TX free interrupt to processor 1, when unmasked.

### 37.4.5 IPCC processor 2 control register (IPCC\_C2CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TXFIE														
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXOIE														
															rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **TXFIE**: Processor 2 transmit channel free interrupt enable

Associated with IPCC\_C2TOC1SR.

1: Enable an unmasked processor 2 transmit channel free to generate a TX free interrupt.

0: Processor 2 TX free interrupt disabled

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **RXOIE**: Processor 2 receive channel occupied interrupt enable

Associated with IPCC\_C1TOC2SR.

1: Enable an unmasked processor 2 receive channel occupied to generate an RX occupied interrupt.

0: Processor 2 RX occupied interrupt disabled

### 37.4.6 IPCC processor 2 mask register (IPCC\_C2MR)

Address offset: 0x014

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CH6 FM	CH5 FM	CH4 FM	CH3 FM	CH2 FM	CH1 FM									
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CH6 OM	CH5 OM	CH4 OM	CH3 OM	CH2 OM	CH1 OM									
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnFM**: Processor 2 transmit channel n free interrupt mask (n = 6 to 1).

Associated with IPCC\_C2TOC1SR.CHnF

1: Transmit channel n free interrupt masked.

0: Transmit channel n free interrupt not masked.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnOM**: Processor 2 receive channel n occupied interrupt mask (n = 6 to 1).

Associated with IPCC\_C1TOC2SR.CHnF

1: Receive channel n occupied interrupt masked.

0: Receive channel n occupied interrupt not masked.

### 37.4.7 IPCC processor 2 status set clear register (IPCC\_C2SCR)

Address offset: 0x018

Reset value: 0x0000 0000

Reading this register always returns 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CH6S	CH5S	CH4S	CH3S	CH2S	CH1S									
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CH6C	CH5C	CH4C	CH3C	CH2C	CH1C									
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnS**: Processor 2 transmit channel n status set (n = 6 to 1).

Associated with IPCC\_C2TOC1SR.CHnF

1: Processor 2 transmit channel n status bit set.

0: No action.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnC**: Processor 2 receive channel n status clear (n = 6 to 1).

Associated with IPCC\_C1TOC2SR.CHnF

1: Processor 2 receive channel n status bit clear.

0: No action.

### 37.4.8 IPCC processor 2 to processor 1 status register (IPCC\_C2TOC1SR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F									

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnF**: Processor 2 transmit to processor 1 receive channel n status flag before masking (n = 6 to 1)

1: Channel occupied, data can be read by the receiving processor 1.

Generates a channel RX occupied interrupt to processor 1, when unmasked.

0: Channel free, data can be written by the sending processor 2.

Generates a channel TX free interrupt to processor 2, when unmasked.

### 37.4.9 IPCC register map

**Table 237. IPCC register map and reset values**

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 38 Hardware semaphore (HSEM)

### 38.1 Introduction

The hardware semaphore block provides 32 (32-bit) register based semaphores.

The semaphores can be used to ensure synchronization between different processes running between different cores. The HSEM provides a non-blocking mechanism to lock semaphores in an atomic way. The following functions are provided:

- Semaphore lock, in two ways:
  - 2-step lock: by writing COREID and PROCID to the semaphore, followed by a read check
  - 1-step lock: by reading the COREID from the semaphore
- Interrupt generation when a semaphore is unlocked
  - Each semaphore may generate an interrupt on one of the interrupt lines
- Semaphore clear protection
  - A semaphore is only unlocked when COREID and PROCID match
- Global semaphore clear per COREID

### 38.2 Main features

The HSEM includes the following features:

- 32 (32-bit) semaphores
- 8-bit PROCID
- 4-bit COREID
- 1 interrupt line per processor
- Lock indication

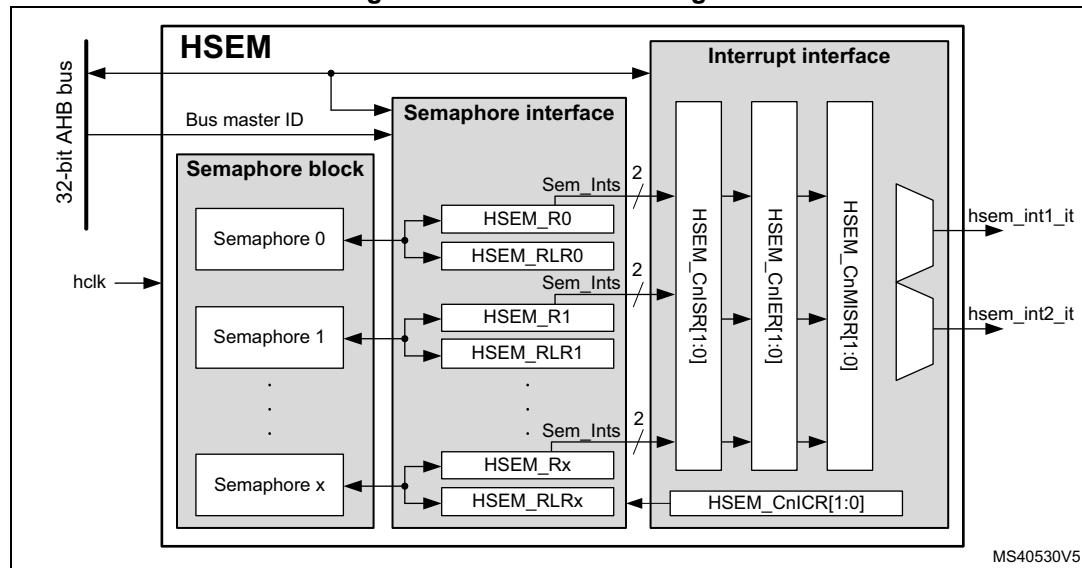
## 38.3 Functional description

### 38.3.1 HSEM block diagram

As shown in [Figure 405](#), the HSEM is based on three sub-blocks:

- the semaphore block containing the semaphore status and IDs
- the semaphore interface block providing AHB access to the semaphore via the HSEM\_Rx and HSEM\_RLRx registers
- the interrupt interface block providing control for the interrupts via HSEM\_CnISR, HSEM\_CnIER, HSEM\_CnMISR, and HSEM\_CnICR registers.

**Figure 405. HSEM block diagram**



### 38.3.2 HSEM internal signals

**Table 238. HSEM internal input/output signals**

Signal name	Signal type	Description
AHB bus	Digital input/output	AHB register access bus
BusMasterID	Digital input	AHB bus master ID
hsem_intn_it	Digital output	Interrupt n line (n = 1 to 2)

### 38.3.3 HSEM lock procedures

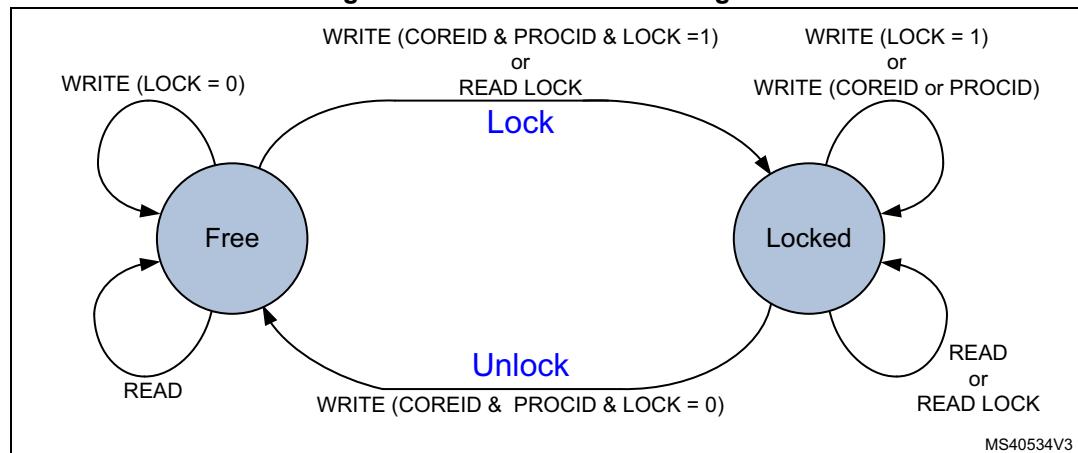
There are two lock procedures, namely 2-step (write) lock and 1-step (read) lock. The two procedures can be used concurrently.

The semaphore is free when its LOCK bit is 0. In this case, the COREID and PROCID are also 0. When the LOCK bit is 1, the semaphore is locked and the COREID indicates which AHB bus master ID has locked it. The PROCID indicates which process of that AHB bus master ID has locked the semaphore.

When write locking a semaphore, the written COREID must match the AHB bus master ID, and the PROCID is written by the AHB bus master software process taking the lock.

When read locking the semaphore, the COREID is taken from the AHB bus master ID, and the PROCID is forced to 0 by hardware. There is no PROCID available with read lock.

**Figure 406. Procedure state diagram**



### 2-step (write) lock procedure

The 2-step lock procedure consists in a write to lock the semaphore, followed by a read to check if the lock has been successful, carried out from the HSEM\_Rx register

- Write semaphore with PROCID and COREID, and LOCK = 1. The COREID data written by software must match the AHB bus master information. i.e. a AHB bus master ID = 1 writes data COREID = 1.  
Lock is put in place when the semaphore is free at write time.
- Read-back the semaphore  
The software checks the lock status, if PROCID and COREID match the written data, then the lock is confirmed.
- Else retry (the semaphore has been locked by another process, AHB bus master ID).

A semaphore can only be locked when it is free.

A semaphore can be locked when the PROCID = 0.

Consecutive write attempts with LOCK = 1 to a locked semaphore are ignored.

### 1-step (read) lock procedure

The 1-step procedure consists in a read to lock and check the semaphore in a single step, carried out from the HSEM\_RLRx register.

- Read lock semaphore with the AHB bus master COREID.
- If read COREID matches and PROCID = 0, then lock is put in place. If COREID matches and PROCID is not 0, this means that another process from the same COREID has locked the semaphore with a 2-step (write) procedure.
- Else retry (the semaphore has been locked by another process, AHB bus master ID).

A semaphore can only be locked when it is free. When read locking a free semaphore, PROCID is 0. Read locking a locked semaphore returns the COREID and PROCID that locked it. All read locks, including the first one that locks the semaphore, return the COREID that locks or locked the semaphore.

*Note:* *The 1-step procedure must not be used when running multiple processes of the same AHB bus master ID. All processes using the same semaphore read the same status. When only one process locks the semaphore, each process of that AHB bus master ID reads the semaphore as locked by itself with the COREID.*

#### 38.3.4 HSEM write/read/read lock register address

For each semaphore, two AHB register addresses are provided, separated in two banks of 32-bit semaphore registers, spaced by a 0x80 address offset.

In the first register address bank the semaphore can be written (locked/unlocked) and read through the HSEM\_Rx registers.

In the second register address bank the semaphore can be read (locked) through the HSEM\_RLRx registers.

#### 38.3.5 HSEM unlock procedures

Unlocking a semaphore is a protected process, to prevent accidental clearing by a AHB bus master ID or by a process not having the semaphore lock right. The procedure consists in writing to the semaphore HSEM\_Rx register with the corresponding COREID and PROCID and LOCK = 0. When unlocked the semaphore, the COREID, and the PROCID are all 0.

When unlocked, an interrupt may be generated to signal the event. To this end, the semaphore interrupt must be enabled.

The unlock procedure consists in a write to the semaphore HSEM\_Rx register with matching COREID regardless on how the semaphore has been locked (1- or 2-step).

- Write semaphore with PROCID, COREID, and LOCK = 0
- If the written data matches the semaphore PROCID and COREID and the AHB bus master ID, the semaphore is unlocked and an interrupt may be generated when enabled, else write is ignored, semaphore remains locked and no interrupt is generated (the semaphore is locked by another process, AHB bus master ID or the written data does not match the AHB bus master signaling).

*Note:* *Different processes of the same AHB bus master ID can write any PROCID value. Preventing other processes of the same AHB bus master ID from unlocking a semaphore must be ensured by software, handling the PROCID correctly.*

### 38.3.6 HSEM COREID semaphore clear

All semaphores locked by a COREID can be unlocked at once by using the HSEM\_CR register. Write COREID and correct KEY value in HSEM\_CR. All locked semaphores with a matching COREID are unlocked, and may generate an interrupt when enabled.

**Note:** *This procedure may be used in case of an incorrect functioning AHB bus master ID, where another AHB bus master can unlock the locked semaphores by writing the COREID of the incorrect functioning processor into the HSEM\_CR register with the correct KEY value. This unlocks all locked semaphores with a matching COREID.*

An interrupt may be generated for the unlocked semaphore(s). To this end, the semaphore interrupt must be enabled in the HSEM\_CnIER registers.

### 38.3.7 HSEM interrupts

An interrupt line hsem\_intn\_it per processor allows each semaphore to generate an interrupt.

An interrupt line provides the following features per semaphore:

- interrupt enable
- interrupt clear
- interrupt status
- masked interrupt status

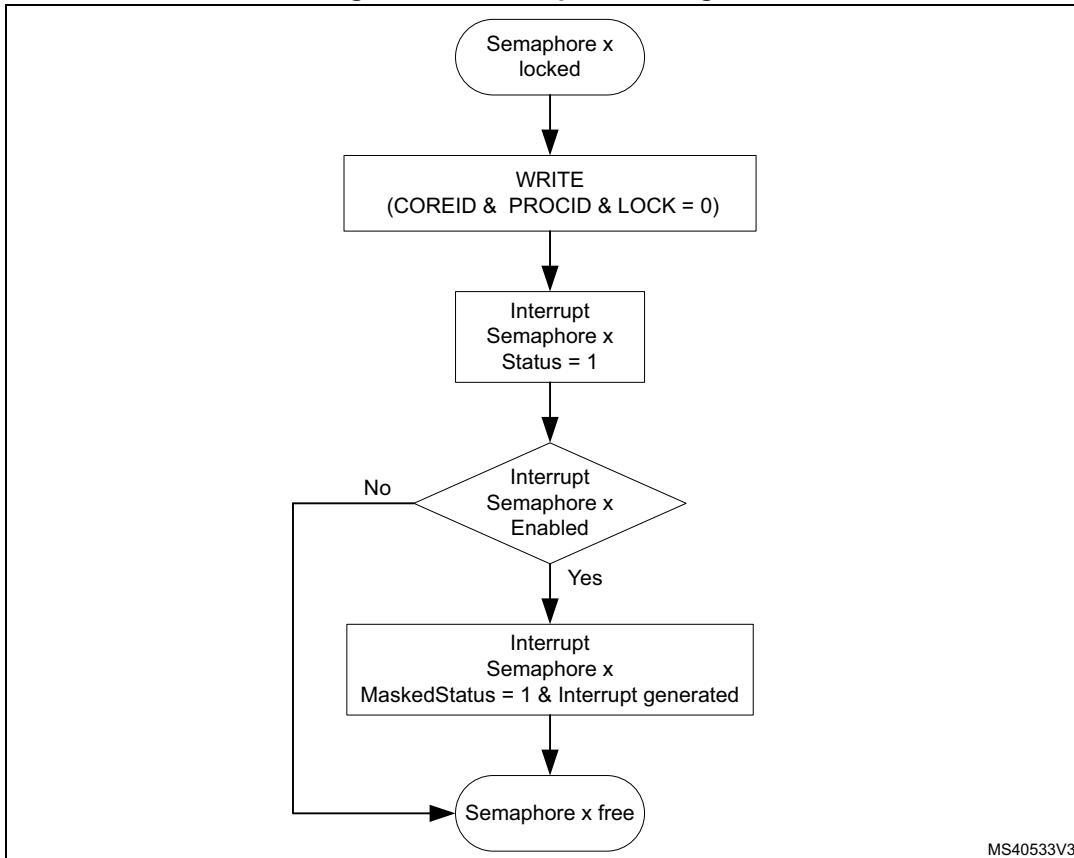
With the interrupt enable (HSEM\_CnIER) the semaphores affecting the interrupt line can be enabled. Disabled (masked) semaphore interrupts do not set the masked interrupt status MISF for that semaphore, and do not generate an interrupt on the interrupt line.

The interrupt clear (HSEM\_CnICR) clears the interrupt status ISF and masked interrupt status MISF of the associated semaphore for the interrupt line.

The interrupt status (HSEM\_CnISR) mirrors the semaphore interrupt status ISF before the enable.

The masked interrupt status (HSEM\_CnMISR) only mirrors the semaphore enabled interrupt status MISF on the interrupt line. All masked interrupt status MISF of the enabled semaphores need to be cleared to clear the interrupt line.

Figure 407. Interrupt state diagram



The procedure to get an interrupt when a semaphore becomes free is described hereafter.

### Try to lock semaphore x

- If the semaphore lock is obtained, no interrupt is needed.
- If the semaphore lock fails:
  - Clear pending semaphore x interrupt status for the interrupt line in HSEM\_CnICR. Re-try to lock the semaphore x again:
    - If the semaphore lock is obtained, no interrupt is needed (semaphore has been freed between first try to lock and clear semaphore interrupt status).
    - If the semaphore lock fails, enable the semaphore x interrupt in HSEM\_CnIER.

### On semaphore x free interrupt, try to lock semaphore x

- If the semaphore lock is obtained:
  - Disable the semaphore x interrupt in HSEM\_CnIER.
  - Clear pending semaphore x interrupt status in HSEM\_CnICR.
- If the semaphore x lock fails:
  - Clear pending semaphore x interrupt status in HSEM\_CnICR.
  - Try again to lock the semaphore x:
    - If the semaphore lock is obtained (semaphore has been freed between first try to lock and semaphore Interrupt status clear), disable the semaphore interrupt in HSEM\_CnIER.

- If the semaphore lock fails, wait for semaphore free interrupt.

**Note:** *An interrupt does not lock the semaphore. After an interrupt, either the AHB bus master or the process must still perform the lock procedure to lock the semaphore.*

It is possible to have multiple AHB bus masters informed by the semaphore free interrupts. Each AHB bus master gets its interrupt, and the first one to react locks the semaphore.

### 38.3.8 AHB bus master ID verification

The HSEM allows only authorized AHB bus master IDs to lock and unlock semaphores.

- The AHB bus master 2-step lock write access to the semaphore HSEM\_Rx register is checked against the valid bus master IDs.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not lock the semaphore.
- The AHB bus master 1-step lock read access from the semaphore HSEM\_RLRx register is checked against the valid bus master IDs.
  - An unauthorized AHB bus master ID read from HSEM\_RLRx returns all 0.
- The semaphore unlock write access to the HSEM\_CR register is checked against the valid bus master IDs. Only the valid bus master IDs can write to the HSEM\_CR register and unlock any of the COREID semaphores.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not clear the COREID semaphores.

*Table 239* details the relation between bus master/processor and COREID.

**Table 239. Authorized AHB bus master IDs**

Bus master 0 (processor1)	Bus master 1 (processor2)
COREID = 4	COREID = 8

**Note:** *Accesses from unauthorized AHB bus master IDs to other registers are granted.*

## 38.4 HSEM registers

Registers must be accessed using word format. Byte and half-word accesses are ignored and have no effect on the semaphores, they generate a bus error.

### 38.4.1 HSEM register semaphore x (HSEM\_Rx)

Address offset:  $0x000 + 0x4 * x$  ( $x = 0$  to  $31$ )

Reset value:  $0x0000\ 0000$

The HSEM\_Rx must be used to perform a 2-step write lock, read back, and for unlocking a semaphore. Only write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
<i>rw</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	COREID[3:0]				PROCID[7:0]							
				<i>rw</i>	<i>rw</i>	<i>rw</i>	<i>rw</i>	<i>rw</i>	<i>rw</i>	<i>rw</i>	<i>rw</i>	<i>rw</i>	<i>rw</i>	<i>rw</i>	<i>rw</i>

Bit 31 **LOCK**: Lock indication

This bit can be written and read by software.

0: On write free semaphore (only when COREID and PROCID match), on read semaphore is free.

1: On write try to lock semaphore, on read semaphore is locked.

Bits 30:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]**: Semaphore COREID

Written by software

- When the semaphore is free and the LOCK bit is at the same time written to 1 and the COREID matches the AHB bus master ID.

- When the semaphore is unlocked (LOCK written to 0 and AHB bus master ID matched COREID, the COREID is cleared to 0).

- When the semaphore is unlocked (LOCK bit written to 0 or AHB bus master ID does not match COREID, the COREID is not affected).

- Write when LOCK bit is already 1 (semaphore locked), the COREID is not affected.

- An authorized read returns the stored COREID value.

Bits 7:0 **PROCID[7:0]**: Semaphore PROCID

Written by software

- When the semaphore is free and the LOCK is written to 1, and the COREID matches the AHB bus master ID, PROCID is set to the written data.

- When the semaphore is unlocked, LOCK written to 0 and AHB bus master ID matched COREID, the PROCID is cleared to 0.

- When the semaphore is unlocked, LOCK bit written to 0 and AHB bus master ID does not match COREID, the PROCID is not affected.

- Write when LOCK bit is already 1 (semaphore locked), the PROCID is not affected.

- An authorized read returns the stored PROCID value.

### 38.4.2 HSEM read lock register semaphore x (HSEM\_RLRx)

Address offset: 0x080 + 0x4 \* x (x = 0 to 31)

Reset value: 0x0000 0000

Accesses the same physical bits as HSEM\_Rx. The HSEM\_RLRx must be used to perform a 1-step read lock. Only read accesses with authorized AHB bus master IDs are granted. Read accesses with unauthorized AHB bus master IDs are discarded and return 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	COREID[3:0]				PROCID[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r

#### Bit 31 **LOCK**: Lock indication

This bit is read only by software at this address.

- When the semaphore is free:

A read with a valid AHB bus master ID locks the semaphore and returns 1.

- When the semaphore is locked:

A read with a valid AHB bus master ID returns 1 (the COREID and PROCID reflect the already locked semaphore information).

Bits 30:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

#### Bits 11:8 **COREID[3:0]**: Semaphore COREID

This field is read only by software at this address.

On a read, when the semaphore is free, the hardware sets the COREID to the AHB bus master ID reading the semaphore. The COREID of the AHB bus master locking the semaphore is read.

On a read when the semaphore is locked, this field returns the COREID of the AHB bus master that has locked the semaphore.

#### Bits 7:0 **PROCID[7:0]**: Semaphore processor ID

This field is read only by software at this address.

- On a read when the semaphore is free:

A read with a valid AHB bus master ID locks the semaphore and hardware sets the PROCID to 0.

- When the semaphore is locked:

A read with a valid AHB bus master ID returns the PROCID of the AHB bus master that has locked the semaphore.

### 38.4.3 HSEM interrupt enable register (HSEM\_CnIER)

Address offset: 0x100 + 0x010 \* (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISE[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
ISE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ISE[31:0]**: Interrupt(n) semaphore x enable bit (x = 0 to 31)

This bit is read and written by software.

0: Interrupt(n) generation for semaphore x disabled (masked)

1: Interrupt(n) generation for semaphore x enabled (not masked)

### 38.4.4 HSEM interrupt clear register (HSEM\_CnICR)

Address offset: 0x104 + 0x010 \* (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISC[31:16]															
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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISC[15:0]															
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 31:0 **ISC[31:0]**: Interrupt(n) semaphore x clear bit (x = 0 to 31)

This bit is written by software, and is always read 0.

0: Interrupt(n) semaphore x status ISFx and masked status MISFx not affected.

1: Interrupt(n) semaphore x status ISFx and masked status MISFx cleared.

### 38.4.5 HSEM interrupt status register (HSEM\_CnISR)

Address offset: 0x108 + 0x010 \* (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ISF[31:0]**: Interrupt semaphore x status bit before enable (mask) (x = 0 to 31)

This bit is set by hardware, and reset only by software. This bit is cleared by software writing the corresponding HSEM\_CnICR bit.

0: Interrupt semaphore x status, no interrupt pending  
1: Interrupt semaphore x status, interrupt pending

### 38.4.6 HSEM interrupt status register (HSEM\_CnMISR)

Address offset: 0x10C + 0x010 \* (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MISF[31:0]**: Masked interrupt(n) semaphore x status bit after enable (mask) (x = 0 to 31)

This bit is set by hardware and read only by software. This bit is cleared by software writing the corresponding HSEM\_CnICR bit. This bit is read as 0 when semaphore x status is masked in HSEM\_CnIER bit x.

0: interrupt(n) semaphore x status after masking not pending  
1: interrupt(n) semaphore x status after masking pending

### 38.4.7 HSEM clear register (HSEM\_CR)

Address offset: 0x140

Reset value: 0x0000 0000

Only write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[15:0]															
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
Res.	Res.	Res.	Res.	COREID[3:0]				Res.							
				w	w	w	w								

Bits 31:16 **KEY[15:0]**: Semaphore clear key

This field can be written by software and is always read 0.

If this key value does not match HSEM\_KEYR.KEY, semaphores are not affected.

If this key value matches HSEM\_KEYR.KEY, all semaphores matching the COREID are cleared to the free state.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]**: COREID of semaphores to be cleared

This field can be written by software and is always read 0.

This field indicates the COREID for which the semaphores are cleared when writing the HSEM\_CR.

Bits 7:0 Reserved, must be kept at reset value.

### 38.4.8 HSEM clear semaphore key register (HSEM\_KEYR)

Address offset: 0x144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[15:0]															
<b>rw</b>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															

Bits 31:16 **KEY[15:0]**: Semaphore clear key

This field can be written and read by software.

Key value to match when clearing semaphores.

Bits 15:0 Reserved, must be kept at reset value.

### 38.4.9 HSEM register map

Table 240. HSEM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0x000	HSEM_R0	LOCK	Res.																																		
	Reset value	0																																			
0x004	HSEM_R1	LOCK	Res.																																		
	Reset value	0																																			
...																																					
0x07C	HSEM_R31	LOCK	LOCK	Res.																																	
	Reset value	0																																			
0x080	HSEM_RLR0	LOCK	LOCK	Res.																																	
	Reset value	0																																			
0x084	HSEM_RLR1	LOCK	LOCK	Res.																																	
	Reset value	0																																			
...																																					
0x0FC	HSEM_RLR31	LOCK	Res.																																		
	Reset value	0																																			
0x100	HSEM_C1IER																																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x104	HSEM_C1ICR																																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x108	HSEM_C1ISR																																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x10C	HSEM_C1MISR																																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x110	HSEM_C2IER																																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x114	HSEM_C2ICR																																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x118	HSEM_C2ISR																																				
	Reset value	0	0	0	0	0	0	0	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 240. HSEM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x11C	<b>HSEM_C2MISR</b>																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x140	<b>HSEM_CR</b>																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x144	<b>HSEM_KEYR</b>																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 39 Universal serial bus full-speed device interface (USB)

### 39.1 Introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB1 bus.

USB suspend/resume are supported, which allows to stop the device clocks for low-power consumption.

### 39.2 USB main features

- USB specification version 2.0 full-speed compliant
- Configurable number of endpoints from 1 to 8
- Dedicated packet buffer memory (SRAM) of 1024 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB 2.0 Link Power Management support
- Battery Charging Specification Revision 1.2 support
- USB connect / disconnect capability (controllable embedded pull-up resistor on USB\_DP line)

### 39.3 USB implementation

*Table 241* describes the USB implementation in the devices.

**Table 241. USB implementation**

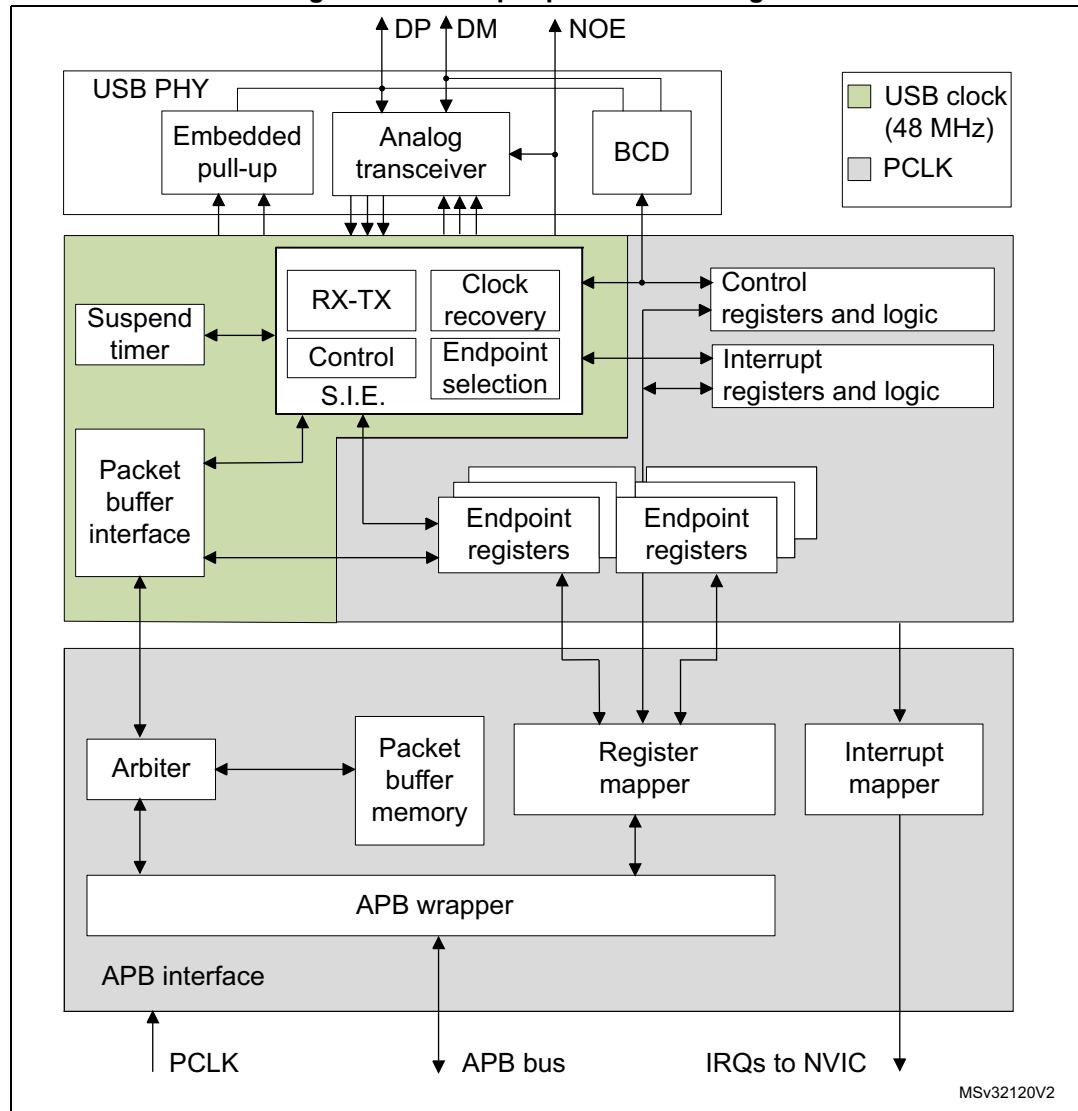
USB features <sup>(1)</sup>	USB
Number of endpoints	8
Size of dedicated packet buffer memory SRAM	1024 bytes
Dedicated packet buffer memory SRAM access scheme	2 x 16 bits / word
USB 2.0 Link Power Management (LPM) support	X
Battery Charging Detection (BCD) support	X
Embedded pull-up resistor on USB_DP line	X

1. X= supported

## 39.4 USB functional description

*Figure 408* shows the block diagram of the USB peripheral.

**Figure 408. USB peripheral block diagram**



The USB peripheral provides an USB-compliant connection between the host PC and the function implemented by the microcontroller. Data transfer between the host PC and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is 1024 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the USB host, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint is associated with a buffer description block indicating where the endpoint-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint is configured) takes

place. The data buffered by the USB peripheral is loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data has been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which allows to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wake-up line to allow the system to immediately restart the normal clock generation and/or support direct clock start/stop.

### 39.4.1 Description of USB blocks

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- USB Physical Interface (USB PHY): This block is maintaining the electrical interface to an external USB host. It contains the differential analog transceiver itself, controllable embedded pull-up resistor (connected to USB\_DP line) and support for Battery Charging Detection (BCD), multiplexed on same USB\_DP and USB\_DM lines. The output enable control signal of the analog transceiver (active low) is provided externally on USB\_NOE. It can be used to drive some activity LED or to provide information about the actual communication direction to some other circuitry.
- Serial Interface Engine (SIE): The functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage. This unit also generates signals according to USB peripheral events, such as Start of Frame (SOF), USB\_Reset, Data errors etc. and to Endpoint related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.
- Timer: This block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- Packet Buffer Interface: This block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the Endpoint registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.

- Endpoint-Related Registers: Each endpoint has an associated register containing the endpoint type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.
- Control Registers: These are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- Interrupt Registers: These contain the Interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

Note:

*\* Endpoint 0 is always used for control transfer in single-buffer mode.*

The USB peripheral is connected to the APB1 bus through an APB1 interface, containing the following blocks:

- Packet Memory: This is the local memory that physically contains the Packet Buffers. It can be used by the Packet Buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the Packet Memory is 1024 bytes, structured as 512 half-words of 16 bits.
- Arbiter: This block accepts memory requests coming from the APB1 bus and from the USB interface. It resolves the conflicts by giving priority to APB1 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB1 transfers of any length are also allowed by this scheme.
- Register Mapper: This block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 16-bit wide half-word set addressed by the APB1.
- APB1 Wrapper: This provides an interface to the APB1 for the memory and register. It also maps the whole USB peripheral in the APB1 address space.
- Interrupt Mapper: This block is used to select how the possible USB events can generate interrupts and map them to the NVIC.

## 39.5 Programming considerations

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

### 39.5.1 Generic USB device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and Isochronous transfers. Apart from system reset, action is always initiated by the USB peripheral, driven by one of the USB events described below.

### 39.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software should perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ( $t_{STARTUP}$  specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the FRES bit in the CNTR register). Clearing the ISTR register then removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

#### USB reset (RESET interrupt)

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral will not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the Enable Function (EF) bit of the USB\_DADDR register and initializing the EP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB\_DADDR register, and configures any other necessary endpoint.

When a RESET interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of reset sequence which triggered the interrupt.

#### Structure and usage of packet buffers

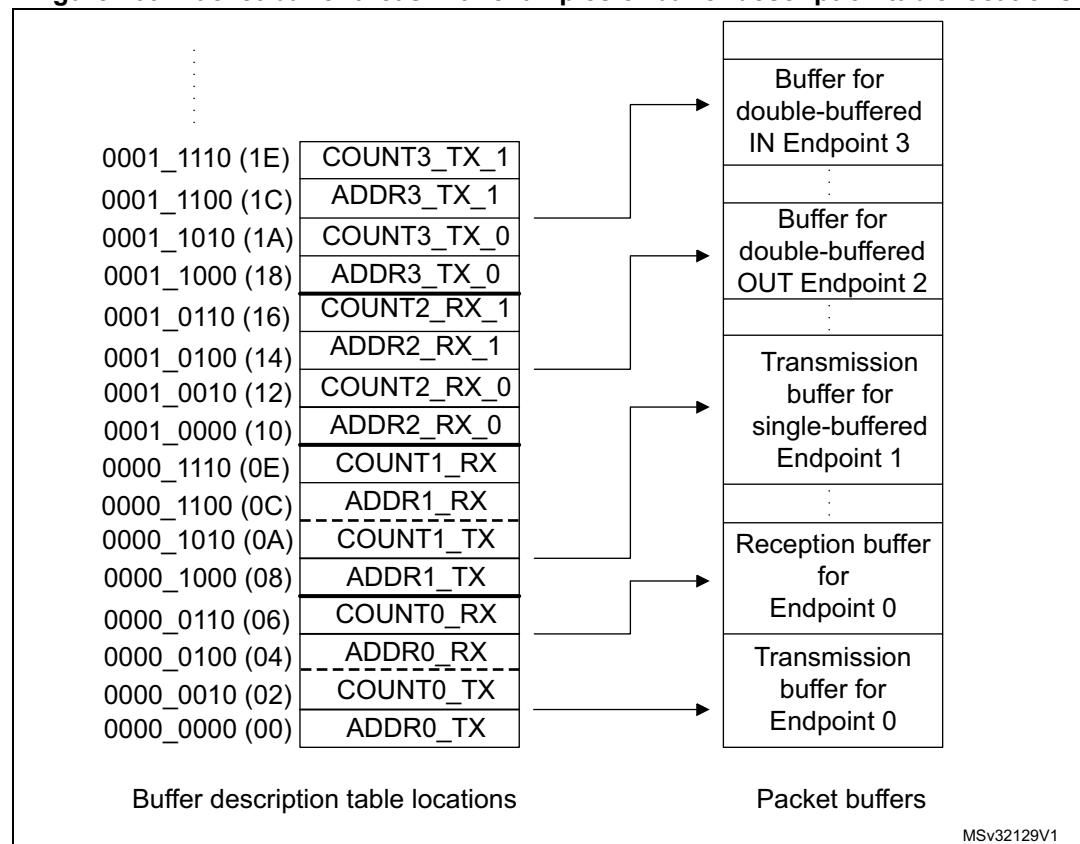
Each bidirectional endpoint may receive or transmit data from/to the host. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request and waits for its acknowledgment. Since the packet buffer memory has to be accessed by the microcontroller also, an arbitration logic takes care of the access conflicts, using half APB1 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both the agents can operate as if the packet memory is a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing

back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB1 bus. Different clock configurations are possible where the APB1 clock frequency can be higher or lower than the USB peripheral one.

*Note:* *Due to USB data rate and packet memory interface requirements, the APB1 clock must have a minimum frequency of 10 MHz to avoid data overrun/underrun problems.*

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory at the address indicated by the USB\_BTABLE register. Each table entry is associated to an endpoint register and it is composed of four 16-bit half-words so that table start address must always be aligned to an 8-byte boundary (the lowest three bits of USB\_BTABLE register are always "000"). Buffer descriptor table entries are described in the [Section 39.6.2: Buffer descriptor table](#). If an endpoint is unidirectional and it is neither an Isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to [Section 39.5.4: Isochronous transfers](#) and [Section 39.5.3: Double-buffered endpoints](#) respectively). The relationship between buffer description table entries and packet buffer areas is depicted in [Figure 409](#).

**Figure 409. Packet buffer areas with examples of buffer description table locations**



Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral will never change the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data are copied to the memory only up to the last available location.

### Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn\_TX/ADDRn\_RX registers so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The EP\_TYPE bits in the USB\_EPnR register must be set according to the endpoint type, eventually using the EP\_KIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STAT\_TX bits in the USB\_EPnR register and COUNTn\_TX must be initialized. For reception, STAT\_RX bits must be set to enable reception and COUNTn\_RX must be written with the allocated buffer size using the BL\_SIZE and NUM\_BLOCK fields. Unidirectional endpoints, except Isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB\_EPnR and locations ADDRn\_TX/ADDRn\_RX, COUNTn\_TX/COUNTn\_RX (respectively), should not be modified by the application software, as the hardware can change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

### IN packets (data transmission)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of ADDRn\_TX and COUNTn\_TX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16 bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (Refer to [Structure and usage of packet buffers on page 1309](#)) and starts sending a DATA0 or DATA1 PID according to USB\_EPnR bit DTOG\_TX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STAT\_TX bits in the USB\_EPnR register.

The ADDR internal register is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn\_TX for COUNTn\_TX/2 half-words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed is used.

On receiving the ACK receipt by the host, the USB\_EPnR register is updated in the following way: DTOG\_TX bit is toggled, the endpoint is made invalid by setting STAT\_TX=10 (NAK) and bit CTR\_TX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP\_ID and DIR bits in the USB\_ISTR register. Servicing of the CTR\_TX event starts clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn\_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STAT\_TX to '11 (VALID) to re-enable transmissions. While the STAT\_TX bits are equal to '10 (NAK), any IN request addressed to that endpoint is NAKed,

indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

### OUT and SETUP packets (data reception)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn\_RX and COUNTn\_RX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn\_RX is stored directly in its internal register ADDR. While COUNT is now reset and the values of BL\_SIZE and NUM\_BLOCK bit fields, which are read within COUNTn\_RX content are used to initialize BUF\_COUNT, an internal 16 bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF\_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but ACK packet is not sent and the ERR bit in USB\_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STAT\_RX in the USB\_EPnR register and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn\_RX for a number of bytes corresponding to the received data packet length, CRC included (i.e. data payload length + 2), or up to the last allocated memory location, as defined by BL\_SIZE and NUM\_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn\_RX location inside the buffer description table entry, leaving unaffected BL\_SIZE and NUM\_BLOCK fields, which normally do not require to be re-written, and the USB\_EPnR register is updated in the following way: DTOG\_RX bit is toggled, the endpoint is made invalid by setting STAT\_RX = '10 (NAK) and bit CTR\_RX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP\_ID and DIR bits in the USB\_ISTR register. The CTR\_RX event is serviced by first determining the transaction type (SETUP bit in the USB\_EPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn\_RX location inside the buffer description table entry related to the endpoint being

processed. After the received data is processed, the application software should set the STAT\_RX bits to '11 (Valid) in the USB\_EPnR, enabling further transactions. While the STAT\_RX bits are equal to '10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

### Control transfers

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOG\_TX and DTOG\_RX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STAT\_TX and STAT\_RX are set to '10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB\_EPnR register at each CTR\_RX event to distinguish normal OUT transactions from SETUP ones. A USB device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction should be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction should be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software will change NAK to VALID, otherwise to STALL. At the same time, if the status stage is an OUT, the STATUS\_OUT (EP\_KIND in the USB\_EPnR register) bit should be set, so that an error is generated if a status transaction is performed with not-zero data. When the status transaction is serviced, the application clears the STATUS\_OUT bit and sets STAT\_RX to VALID (to accept a new command) and STAT\_TX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic doesn't allow a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STAT\_RX bits are set to '01 (STALL) or '10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued CTR\_RX request not yet acknowledged by the application (i.e. CTR\_RX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the CTR\_RX interrupt.

### 39.5.3 Double-buffered endpoints

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it will answer with a NAK handshake and the host PC will issue the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called 'double-buffering' can be used with bulk endpoints.

When 'double-buffering' is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both 'transmission' and 'reception' packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a 'reception' double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a 'transmission' double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB\_EPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from '00 (Disabled): STAT\_RX if the double-buffered bulk endpoint is enabled for reception, STAT\_TX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB\_EPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOG\_RX (bit 14 of USB\_EPnR register) for 'reception' double-buffered bulk endpoints or DTOG\_TX (bit 6 of USB\_EPnR register) for 'transmission' double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral should know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB\_EPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW\_BUF. In the following table the correspondence between USB\_EPnR register bits and DTOG/SW\_BUF definition is explained, for the cases of 'transmission' and 'reception' double-buffered bulk endpoints.

**Table 242. Double-buffering buffer flag definition**

Buffer flag	'Transmission' endpoint	'Reception' endpoint
DTOG	DTOG_TX (USB_EPnR bit 6)	DTOG_RX (USB_EPnR bit 14)
SW_BUF	USB_EPnR bit 14	USB_EPnR bit 6

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW\_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

**Table 243. Bulk double-buffering memory buffers usage**

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
IN	0	1	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations.
	1	0	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	0	0	None <sup>(1)</sup>	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	1	1	None <sup>(1)</sup>	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations.
OUT	0	1	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.
	1	0	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	0	0	None <sup>(1)</sup>	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	1	1	None <sup>(1)</sup>	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by:

- Writing EP\_TYPE bit field at '00 in its USB\_EPnR register, to define the endpoint as a bulk, and
- Setting EP\_KIND bit at '1 (DBL\_BUF), in the same register.

The application software is responsible for DTOG and SW\_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL\_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL\_BUF remain set. At the end of each transaction the CTR\_RX or CTR\_TX bit of the addressed endpoint USB\_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB\_EPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after

DBL\_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains '11 (Valid). However, as the token packet of a new transaction is received, the actual endpoint status is masked as '10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW\_BUF having the same value, see [Table 243 on page 1315](#)). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW\_BUF bit, writing '1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control will take place and the actual transfer rate is limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from '11 (Valid) into the STAT bit pair of the related USB\_EPnR register. In this case, the USB peripheral will always use the programmed endpoint status, regardless of the buffer usage condition.

### 39.5.4 Isochronous transfers

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as 'Isochronous'. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be 'isochronous' during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for Isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, Isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The Isochronous behavior for an endpoint is selected by setting the EP\_TYPE bits at '10 in its USB\_EPnR register; since there is no handshake phase the only legal values for the STAT\_RX/STAT\_TX bit pairs are '00 (Disabled) and '11 (Valid), any other value will produce results not compliant to USB standard. Isochronous endpoints implement double-buffering to ease application software development, using both 'transmission' and 'reception' packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOG\_RX for 'reception' isochronous endpoints, DTOG\_TX for 'transmission' isochronous endpoints, both in the related USB\_EPnR register) according to [Table 244](#).

**Table 244. Isochronous memory buffers usage**

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
IN	0	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.
	1	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.
OUT	0	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.
	1	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.

As it happens with double-buffered bulk endpoints, the USB\_EPnR registers used to implement Isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have Isochronous endpoints enabled both for reception and transmission, two USB\_EPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the CTR\_RX or CTR\_TX bit of the addressed endpoint USB\_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB\_EPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for Isochronous transfers due to the lack of handshake phase, the endpoint remains always '11 (Valid). CRC errors or buffer-overrun conditions occurring during Isochronous OUT transfers are anyway considered as correct transactions and they always trigger an CTR\_RX event. However, CRC errors will anyway set the ERR bit in the USB\_ISTR register to notify the software of the possible data corruption.

### 39.5.5 Suspend/Resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to '1' in USB\_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the FSUSP bit in the USB\_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set LP\_MODE bit in USB\_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to insure that this process does not take more than 10 ms when the wakening event is an USB reset sequence (See “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the LP\_MODE bit in USB\_CNTR register asynchronously. Even if this event can trigger an WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure should address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear FSUSP bit of USB\_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB\_FNR register can be used according to [Table 245](#), which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the Idle bus state; moreover at the end of a reset sequence the RESET bit in USB\_ISTR register is set to 1, issuing an interrupt if enabled, which should be handled as usual.

**Table 245. Resume event detection**

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode

**Table 245. Resume event detection (continued)**

[RXDP,RXDM] status	Wake-up event	Required resume software action
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (e.g. a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the RESUME bit in the USB\_CNTR register to '1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the RESUME bit is clear, the resume sequence is completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB\_FNR register.

*Note: The RESUME bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the FSUSP bit in USB\_CNTR register to 1.*

## 39.6 USB and USB SRAM registers

The USB peripheral registers can be divided into the following groups:

- Common Registers: Interrupt and Control registers
- Endpoint Registers: Endpoint configuration and status

The USB SRAM registers cover:

- Buffer Descriptor Table: Location of packet memory used to locate data buffers (see [Section 2.2: Memory organization](#) to find USB SRAM base address).

All register addresses are expressed as offsets with respect to the USB peripheral registers base address, except the buffer descriptor table locations, which starts at the USB SRAM base address offset by the value specified in the USB\_BTABLE register.

Refer to [Section 1.2 on page 61](#) for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 39.6.1 Common registers

These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.

#### USB control register (USB\_CNTR)

Address offset: 0x40

Reset value: 0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR M	PMAOVR M	ERR M	WKUP M	SUSP M	RESET M	SOF M	ESOF M	L1REQ M	Res	L1RESU ME	RE SUME	F SUSP	LP MODE	PDW N	F RES
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bit 15 **CTRM**: Correct transfer interrupt mask

0: Correct Transfer (CTR) Interrupt disabled.

1: CTR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.

Bit 14 **PMAOVRM**: Packet memory area over / underrun interrupt mask

0: PMAOVR Interrupt disabled.

1: PMAOVR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.

Bit 13 **ERRM**: Error interrupt mask

0: ERR Interrupt disabled.

1: ERR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.

Bit 12 **WKUPM**: Wake-up interrupt mask

0: WKUP Interrupt disabled.

1: WKUP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.

- Bit 11 **SUSPM:** Suspend mode interrupt mask  
 0: Suspend Mode Request (SUSP) Interrupt disabled.  
 1: SUSP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 10 **RESETM:** USB reset interrupt mask  
 0: RESET Interrupt disabled.  
 1: RESET Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 9 **SOFM:** Start of frame interrupt mask  
 0: SOF Interrupt disabled.  
 1: SOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 8 **ESOFM:** Expected start of frame interrupt mask  
 0: Expected Start of Frame (ESOF) Interrupt disabled.  
 1: ESOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 7 **L1REQM:** LPM L1 state request interrupt mask  
 0: LPM L1 state request (L1REQ) Interrupt disabled.  
 1: L1REQ Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 **L1RESUME:** LPM L1 Resume request  
 The microcontroller can set this bit to send a LPM L1 Resume signal to the host. After the signaling ends, this bit is cleared by hardware.
- Bit 4 **RESUME:** Resume request  
 The microcontroller can set this bit to send a Resume signal to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the Host PC is ready to drive the resume sequence up to its end.
- Bit 3 **FSUSP:** Force suspend  
 Software must set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms.  
 0: No effect.  
 1: Enter suspend mode. Clocks and static power dissipation in the analog transceiver are left unaffected. If suspend power consumption is a requirement (bus-powered device), the application software should set the LP\_MODE bit after FSUSP as explained below.
- Bit 2 **LP\_MODE:** Low-power mode  
 This mode is used when the suspend-mode power constraints require that all static power dissipation is avoided, except the one required to supply the external pull-up resistor. This condition should be entered when the application is ready to stop all system clocks, or reduce their frequency in order to meet the power consumption requirements of the USB suspend condition. The USB activity during the suspend mode (WKUP event) asynchronously resets this bit (it can also be reset by software).  
 0: No Low-power mode.  
 1: Enter Low-power mode.

Bit 1 **PDWN**: Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.

- 0: Exit Power Down.  
1: Enter Power down mode.

Bit 0 **FRES**: Force USB Reset

- 0: Clear USB reset.  
1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.

**USB interrupt status register (USB\_ISTR)**

Address offset: 0x44

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
CTR	PMA OVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	L1REQ	Res.	Res.	DIR	EP_ID[3:0]					
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			r	r	r	r	r		

This register contains the status of all the interrupt sources allowing application software to determine, which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB\_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, will perform all necessary actions, and finally it will clear the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line is kept high again. If several bits are set simultaneously, only a single interrupt is generated.

Endpoint transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB\_CNTR is set. An endpoint dedicated interrupt condition is activated independently from the CTRM bit in the USB\_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB\_EPnR register (the CTR bit is actually a read only bit). For endpoint-related interrupts, the software can use the Direction of Transaction (DIR) and EP\_ID read-only bits to identify, which endpoint made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB\_ISTR events by specifying the order in which software checks USB\_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt is requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with '0 (these bits can only be cleared by software). Read-modify-write cycles should be avoided because between the read and the write operations another bit

could be set by the hardware and the next write will clear it before the microprocessor has the time to serve the event.

The following describes each bit in detail:

**Bit 15 CTR: Correct transfer**

This bit is set by the hardware to indicate that an endpoint has successfully completed a transaction; using DIR and EP\_ID bits software can determine which endpoint requested the interrupt. This bit is read-only.

**Bit 14 PMAOVR: Packet memory area over / underrun**

This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host will retry the transaction. The PMAOVR interrupt should never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during Isochronous transfers (no Isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only '0 can be written and writing '1 has no effect.

**Bit 13 ERR: Error**

This flag is set whenever one of the errors listed below has occurred:

NANS: No ANSwer. The timeout for a host response has expired.

CRC: Cyclic Redundancy Check error. One of the received CRCs, either in the token or in the data, was wrong.

BST: Bit Stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.

FVIO: Framing format Violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).

The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (e.g. loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only '0 can be written and writing '1 has no effect.

**Bit 12 WKUP: Wake-up**

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the LP\_MODE bit in the CTR register and activates the USB\_WAKEUP line, which can be used to notify the rest of the device (e.g. wake-up unit) about the start of the resume process. This bit is read/write but only '0 can be written and writing '1 has no effect.

**Bit 11 SUSP: Suspend mode request**

This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (FSUSP=1) until the end of resume sequence. This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 10 **RESET**: USB reset request

Set when the USB peripheral detects an active USB RESET signal at its inputs. The USB peripheral, in response to a RESET, just resets its internal protocol state machine, generating an interrupt if RESETM enable bit in the USB\_CNTR register is set. Reception and transmission are disabled until the RESET bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RESET interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.

This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 9 **SOF**: Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine may monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB\_FNR register which is updated at the SOF packet reception (this could be useful for isochronous applications). This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 8 **ESOF**: Expected start of frame

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the device does not receive it properly, the Suspend Timer issues this interrupt. If three consecutive ESOF interrupts are generated (i.e. three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the Suspend Timer is not yet locked. This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 7 **L1REQ**: LPM L1 state request

This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only '0 can be written and writing '1 has no effect.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **DIR**: Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.

If DIR bit=0, CTR\_TX bit is set in the USB\_EPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).

If DIR bit=1, CTR\_RX bit or both CTR\_TX/CTR\_RX are set in the USB\_EPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed.

This information can be used by the application software to access the USB\_EPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **EP\_ID[3:0]: Endpoint Identifier**

These bits are written by the hardware according to the endpoint number, which generated the interrupt request. If several endpoint transactions are pending, the hardware writes the endpoint identifier related to the endpoint having the highest priority defined in the following way: Two endpoint sets are defined, in order of priority: Isochronous and double-buffered bulk endpoints are considered first and then the other endpoints are examined. If more than one endpoint from the same set is requesting an interrupt, the EP\_ID bits in USB\_ISTR register are assigned according to the lowest requesting endpoint register, EP0R having the highest priority followed by EP1R and so on. The application software can assign a register to each endpoint according to this priority scheme, so as to order the concurring endpoint requests in a suitable way. These bits are read only.

**USB frame number register (USB\_FNR)**

Address offset: 0x48

Reset value: 0x0XXX where X is undefined

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]													
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r			

Bit 15 **RXDP: Receive data + line status**

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 14 **RXDM: Receive data - line status**

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 13 **LCK: Locked**

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]: Lost SOF**

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]: Frame number**

This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for Isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

**USB device address (USB\_DADDR)**

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EF	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved

Bit 7 **EF**: Enable function

This bit is set by the software to enable the USB device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at '0' no transactions are handled, irrespective of the settings of USB\_EPnR registers.

Bits 6:0 **ADD[6:0]**: Device address

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the Endpoint Address (EA) field in the associated USB\_EPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

### Buffer table address (USB\_BTABLE)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BTABLE[15:3]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 15:3 **BTABLE[15:3]**: Buffer table

These bits contain the start address of the buffer allocation table inside the dedicated packet memory. This table describes each endpoint buffer location and size and it must be aligned to an 8 byte boundary (the 3 least significant bits are always '0'). At the beginning of every transaction addressed to this device, the USB peripheral reads the element of this table related to the addressed endpoint, to get its buffer start location and the buffer size (Refer to [Structure and usage of packet buffers on page 1309](#)).

Bits 2:0 Reserved, forced by hardware to 0.

### LPM control and status register (USB\_LPMCSR)

Address offset: 0x54

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BESL[3:0]				REM WAKE	Res.	LPM ACK	LPM EN							
								r	r	r	r	r		rw	rw

Bits 15:8 Reserved, must be kept at reset value.

**Bits 7:4 BESL[3:0]: BESL value**

These bits contain the BESL value received with last ACKed LPM Token

**Bit 3 REMWAKE: bRemoteWake value**

This bit contains the bRemoteWake value received with last ACKed LPM Token

**Bit 2 Reserved**

**Bit 1 LPMACK: LPM Token acknowledge enable**

0: the valid LPM Token is NYET.

1: the valid LPM Token is ACK.

The NYET/ACK is returned only on a successful LPM transaction:

No errors in both the EXT token and the LPM token (else ERROR)

A valid bLinkState = 0001B (L1) is received (else STALL)

**Bit 0 LPMEN: LPM support enable**

This bit is set by the software to enable the LPM support within the USB device. If this bit is at '0' no LPM transactions are handled.

## Battery charging detector (USB\_BCDR)

Address offset: 0x58

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPPU	Res.	PS2 DET	SDET	PDET	DC DET	SDEN	PDEN	DCD EN	BCD EN						
rw								r	r	r	r	rw	rw	rw	rw

**Bit 15 DPPU: DP pull-up control**

This bit is set by software to enable the embedded pull-up on the DP line. Clearing it to '0' can be used to signalize disconnect to the host when needed by the user software.

Bits 14:8 Reserved, must be kept at reset value.

**Bit 7 PS2DET: DM pull-up detection status**

This bit is active only during PD and gives the result of comparison between DM voltage level and  $V_{LGC}$  threshold. In normal situation, the DM level should be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, ACA, CDP or DCP).

1: PS2 port or proprietary charger detected.

**Bit 6 SDET: Secondary detection (SD) status**

This bit gives the result of SD.

0: CDP detected.

1: DCP detected.

**Bit 5 PDET: Primary detection (PD) status**

This bit gives the result of PD.

0: no BCD support detected (connected to SDP or proprietary device).

1: BCD support detected (connected to ACA, CDP or DCP).

Bit 4 **DCDET**: Data contact detection (DCD) status

This bit gives the result of DCD.

0: data lines contact not detected.

1: data lines contact detected.

Bit 3 **SDEN**: Secondary detection (SD) mode enable

This bit is set by the software to put the BCD into SD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 2 **PDEN**: Primary detection (PD) mode enable

This bit is set by the software to put the BCD into PD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 1 **DCDEN**: Data contact detection (DCD) mode enable

This bit is set by the software to put the BCD into DCD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 0 **BCDEN**: Battery charging detector (BCD) enable

This bit is set by the software to enable the BCD support within the USB device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD should be placed in OFF mode by clearing this bit to '0' in order to allow the normal USB operation.

## Endpoint-specific registers

The number of these registers varies according to the number of endpoints that the USB peripheral is designed to handle. The USB peripheral supports up to 8 bidirectional endpoints. Each USB device must support a control endpoint whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint number value. For each endpoint, an USB\_EPnR register is available to store the endpoint specific information.

### USB endpoint n register (USB\_EPnR), n=[0..7]

Address offset: 0x00 to 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP_TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

They are also reset when an USB reset is received from the USB bus or forced through bit FRES in the CTR register, except the CTR\_RX and CTR\_TX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint has its USB\_EPnR register where *n* is the endpoint identifier.

Read-modify-write cycles on these registers should be avoided because between the read and the write operations some bits could be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an 'invariant' value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their 'invariant' value.

Bit 15 **CTR\_RX**: Correct transfer for reception

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB\_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only '0' can be written, writing '1' has no effect.

Bit 14 **DTOG\_RX**: Data toggle, for reception transfers

If the endpoint is not Isochronous, this bit contains the expected value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent to the USB host, following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to [Section 39.5.3: Double-buffered endpoints](#)).

If the endpoint is Isochronous, this bit is used only to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to [Section 39.5.4: Isochronous transfers](#)). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes '0', the value of DTOG\_RX remains unchanged, while writing '1' makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STAT\_RX [1:0]**: Status bits, for reception transfers

These bits contain information about the endpoint status, which are listed in [Table 246: Reception status encoding on page 1331](#). These bits can be toggled by software to initialize their value. When the application software writes '0', the value remains unchanged, while writing '1' makes the bit value toggle. Hardware sets the STAT\_RX bits to NAK when a correct transfer has occurred (CTR\_RX=1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledge a new transaction.

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to [Section 39.5.3: Double-buffered endpoints](#)).

If the endpoint is defined as Isochronous, its status can be only "VALID" or "DISABLED", so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT\_RX bits to 'STALL' or 'NAK' for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing '1'.

Bit 11 **SETUP**: Setup transaction completed

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (CTR\_RX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while CTR\_RX bit is at 1; its state changes when CTR\_RX is at 0. This bit is read-only.

Bits 10:9 **EP\_TYPE[1:0]**: Endpoint type

These bits configure the behavior of this endpoint as described in [Table 247: Endpoint type encoding on page 1331](#). Endpoint 0 must always be a control endpoint and each USB function must have at least one control endpoint which has address 0, but there may be other control endpoints if required. Only control endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint is defined as NAK, the USB peripheral will not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint is defined as STALL in the receive direction, then the SETUP packet is accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint is a control one.

Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EP\_KIND configuration bit.

The usage of Isochronous endpoints is explained in [Section 39.5.4: Isochronous transfers](#)

Bit 8 **EP\_KIND**: Endpoint kind

The meaning of this bit depends on the endpoint type configured by the EP\_TYPE bits. [Table 248](#) summarizes the different meanings.

DBL\_BUF: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in [Section 39.5.3: Double-buffered endpoints](#).

STATUS\_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered 'STALL' instead of 'ACK'. This bit may be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS\_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **CTR\_TX**: Correct Transfer for transmission

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB\_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only '0' can be written.

Bit 6 **DTOG\_TX**: Data Toggle, for transmission transfers

If the endpoint is non-isochronous, this bit contains the required value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to [Section 39.5.3: Double-buffered endpoints](#))

If the endpoint is Isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to [Section 39.5.4: Isochronous transfers](#)). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for Isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes '0, the value of DTOG\_TX remains unchanged, while writing '1 makes the bit value toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 **STAT\_TX [1:0]**: Status bits, for transmission transfers

These bits contain the information about the endpoint status, listed in [Table 249](#). These bits can be toggled by the software to initialize their value. When the application software writes '0, the value remains unchanged, while writing '1 makes the bit value toggle. Hardware sets the STAT\_TX bits to NAK, when a correct transfer has occurred (CTR\_TX=1) corresponding to a IN or SETUP (control only) transaction addressed to this endpoint. It then waits for the software to prepare the next set of data to be transmitted.

Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to [Section 39.5.3: Double-buffered endpoints](#)).

If the endpoint is defined as Isochronous, its status can only be "VALID" or "DISABLED". Therefore, the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT\_TX bits to 'STALL' or 'NAK' for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing '1.

Bits 3:0 **EA[3:0]**: Endpoint address

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

**Table 246. Reception status encoding**

<b>STAT_RX[1:0]</b>	<b>Meaning</b>
00	<b>DISABLED</b> : all reception requests addressed to this endpoint are ignored.
01	<b>STALL</b> : the endpoint is stalled and all reception requests result in a STALL handshake.
10	<b>NAK</b> : the endpoint is naked and all reception requests result in a NAK handshake.
11	<b>VALID</b> : this endpoint is enabled for reception.

**Table 247. Endpoint type encoding**

<b>EP_TYPE[1:0]</b>	<b>Meaning</b>
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

**Table 248. Endpoint kind meaning**

<b>EP_TYPE[1:0]</b>		<b>EP_KIND meaning</b>
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	Not used
11	INTERRUPT	Not used

**Table 249. Transmission status encoding**

STAT_TX[1:0]	Meaning
00	<b>DISABLED</b> : all transmission requests addressed to this endpoint are ignored.
01	<b>STALL</b> : the endpoint is stalled and all transmission requests result in a STALL handshake.
10	<b>NAK</b> : the endpoint is naked and all transmission requests result in a NAK handshake.
11	<b>VALID</b> : this endpoint is enabled for transmission.

### 39.6.2 Buffer descriptor table

**Note:** *The buffer descriptor table is located inside the packet buffer memory in the separate "USB SRAM" address space.*

Although the buffer descriptor table is located inside the packet buffer memory ("USB SRAM" area), its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

The first packet memory location is located at USB SRAM base address. The buffer descriptor table entry associated with the USB\_EPnR registers is described below. The packet memory should be accessed only by byte (8-bit) or half-word (16-bit) accesses. Word (32-bit) accesses are not allowed.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in [Structure and usage of packet buffers on page 1309](#).

#### Transmission buffer address n (USB\_ADDRn\_TX)

Address offset: [USB\_BTABLE] + n\*8

**Note:** *In case of double-buffered or isochronous endpoints in the IN direction, this address location is referred to as USB\_ADDRn\_TX\_0.*

*In case of double-buffered or isochronous endpoints in the OUT direction, this address location is used for USB\_ADDRn\_RX\_0.*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRn_TX[15:1]															-
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

##### Bits 15:1 ADDRn\_TX[15:1]: Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint associated with the USB\_EPnR register at the next IN token addressed to it.

Bit 0 Must always be written as '0' since packet memory is half-word wide and all packet buffers must be half-word aligned.

#### Transmission byte count n (USB\_COUNTn\_TX)

Address offset: [USB\_BTABLE] + n\*8 + 2

**Note:** *In case of double-buffered or isochronous endpoints in the IN direction, this address location is referred to as USB\_COUNTn\_TX\_0.*

*In case of double-buffered or isochronous endpoints in the OUT direction, this address location is used for USB\_COUNTn\_RX\_0.*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
Res.	Res.	Res.	Res.	Res.	Res.	COUNTn_TX[9:0]														
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					

Bits 15:10 These bits are not used since packet size is limited by USB specifications to 1023 bytes. Their value is not considered by the USB peripheral.

Bits 9:0 **COUNTn\_RX[9:0]**: Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint associated with the USB\_EPnR register at the next IN token addressed to it.

### Reception buffer address n (USB\_ADDRn\_RX)

Address offset: [USB\_BTABLE] + n\*8 + 4

**Note:**

*In case of double-buffered or isochronous endpoints in the OUT direction, this address location is referred to as USB\_ADDRn\_RX\_1.*

*In case of double-buffered or isochronous endpoints in the IN direction, this address location is used for USB\_ADDRn\_RX\_1.*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRn_RX[15:1]															-
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	-

Bits 15:1 **ADDRn\_RX[15:1]**: Reception buffer address

These bits point to the starting address of the packet buffer, which will contain the data received by the endpoint associated with the USB\_EPnR register at the next OUT/SETUP token addressed to it.

Bit 0 This bit must always be written as '0' since packet memory is half-word wide and all packet buffers must be half-word aligned.

### Reception byte count n (USB\_COUNTn\_RX)

Address offset: [USB\_BTABLE] + n\*8 + 6

**Note:**

*In case of double-buffered or isochronous endpoints in the OUT direction, this address location is referred to as USB\_COUNTn\_RX\_1.*

*In case of double-buffered or isochronous endpoints in the IN direction, this address location is used for USB\_COUNTn\_RX\_1.*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNTn_RX[9:0]															
BLSIZE	NUM_BLOCK[4:0]					r	r	r	r	r	r	r	r	r	r
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint descriptor and it is normally defined during the

enumeration process according to its maxPacketSize parameter value (See “Universal Serial Bus Specification”).

Bit 15 **BL\_SIZE**: Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BL\_SIZE=0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BL\_SIZE=1, the memory block is 32-byte large, which allows to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 14:10 **NUM\_BLOCK[4:0]**: Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BL\_SIZE value as illustrated in [Table 250](#).

Bits 9:0 **COUNTn\_RX[9:0]**: Reception byte count

These bits contain the number of bytes received by the endpoint associated with the USB\_EPnR register during the last OUT/SETUP transaction addressed to it.

**Table 250. Definition of allocated buffer memory**

Value of <b>NUM_BLOCK[4:0]</b>	Memory allocated when <b>BL_SIZE=0</b>	Memory allocated when <b>BL_SIZE=1</b>
0 ('00000)	Not allowed	32 bytes
1 ('00001)	2 bytes	64 bytes
2 ('00010)	4 bytes	96 bytes
3 ('00011)	6 bytes	128 bytes
...	...	...
14 ('01110)	28 bytes	480 bytes
15 ('01111)	30 bytes	
16 ('10000)	32 bytes	
...	...	...
29 ('11101)	58 bytes	
30 ('11110)	60 bytes	
31 ('11111)	62 bytes	N/A

### 39.6.3 USB register map

The table below provides the USB register map and reset values.

Table 251. USB register map and reset values

Offset	Register	Reset	31
0x00	<b>USB_EP0R</b>	Res.	Res.
		Reset value	30
0x04	<b>USB_EP1R</b>	Res.	Res.
		Reset value	29
0x08	<b>USB_EP2R</b>	Res.	Res.
		Reset value	28
0x0C	<b>USB_EP3R</b>	Res.	Res.
		Reset value	27
0x10	<b>USB_EP4R</b>	Res.	Res.
		Reset value	26
0x14	<b>USB_EP5R</b>	Res.	Res.
		Reset value	25
0x18	<b>USB_EP6R</b>	Res.	Res.
		Reset value	24
0x1C	<b>USB_EP7R</b>	Res.	Res.
		Reset value	23
0x20-0x3F	<b>USB_CNTR</b>	Res.	Res.
		Reset value	22
0x40	<b>USB_ISTR</b>	Res.	Res.
		Reset value	21
0x44	<b>USB_FNR</b>	Res.	Res.
		Reset value	20
0x48	<b>USB_DADDR</b>	Res.	Res.
		Reset value	19
0x4C	<b>USB_DADDR</b>	Res.	Res.
		Reset value	18
Reserved			
0x40	<b>USB_CNTR</b>	Res.	Res.
		Reset value	17
0x44	<b>USB_ISTR</b>	Res.	Res.
		Reset value	16
0x48	<b>USB_FNR</b>	Res.	Res.
		Reset value	15
0x4C	<b>USB_DADDR</b>	Res.	Res.
		Reset value	14
FN[10:0]			
0x40	<b>USB_CNTR</b>	Res.	Res.
		Reset value	13
0x44	<b>USB_ISTR</b>	Res.	Res.
		Reset value	12
0x48	<b>USB_FNR</b>	Res.	Res.
		Reset value	11
0x4C	<b>USB_DADDR</b>	Res.	Res.
		Reset value	10
ADD[6:0]			
0x40	<b>USB_CNTR</b>	Res.	Res.
		Reset value	9
0x44	<b>USB_ISTR</b>	Res.	Res.
		Reset value	8
0x48	<b>USB_FNR</b>	Res.	Res.
		Reset value	7
0x4C	<b>USB_DADDR</b>	Res.	Res.
		Reset value	6
STAT_TX[1:0]			
0x40	<b>USB_CNTR</b>	Res.	Res.
		Reset value	5
0x44	<b>USB_ISTR</b>	Res.	Res.
		Reset value	4
0x48	<b>USB_FNR</b>	Res.	Res.
		Reset value	3
0x4C	<b>USB_DADDR</b>	Res.	Res.
		Reset value	2
STAT_TX[1:0]			
0x40	<b>USB_CNTR</b>	Res.	Res.
		Reset value	1
0x44	<b>USB_ISTR</b>	Res.	Res.
		Reset value	0
FRES			

Table 251. USB register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x50	<b>USB_BTABLE</b>	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
0x54	<b>USB_LPMCSR</b>	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
0x58	<b>USB_BCDR</b>	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
	Reset value	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 40 Clock recovery system (CRS)

### 40.1 Introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means for oscillator output frequency evaluation, based on comparison with a selectable synchronization signal. It is capable of doing automatic adjustment of oscillator trimming based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In such case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, which is sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from the LSE oscillator output or it can be generated by user software.

### 40.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity:
  - LSE oscillator output
  - packet reception
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster start-up convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

### 40.3 CRS implementation

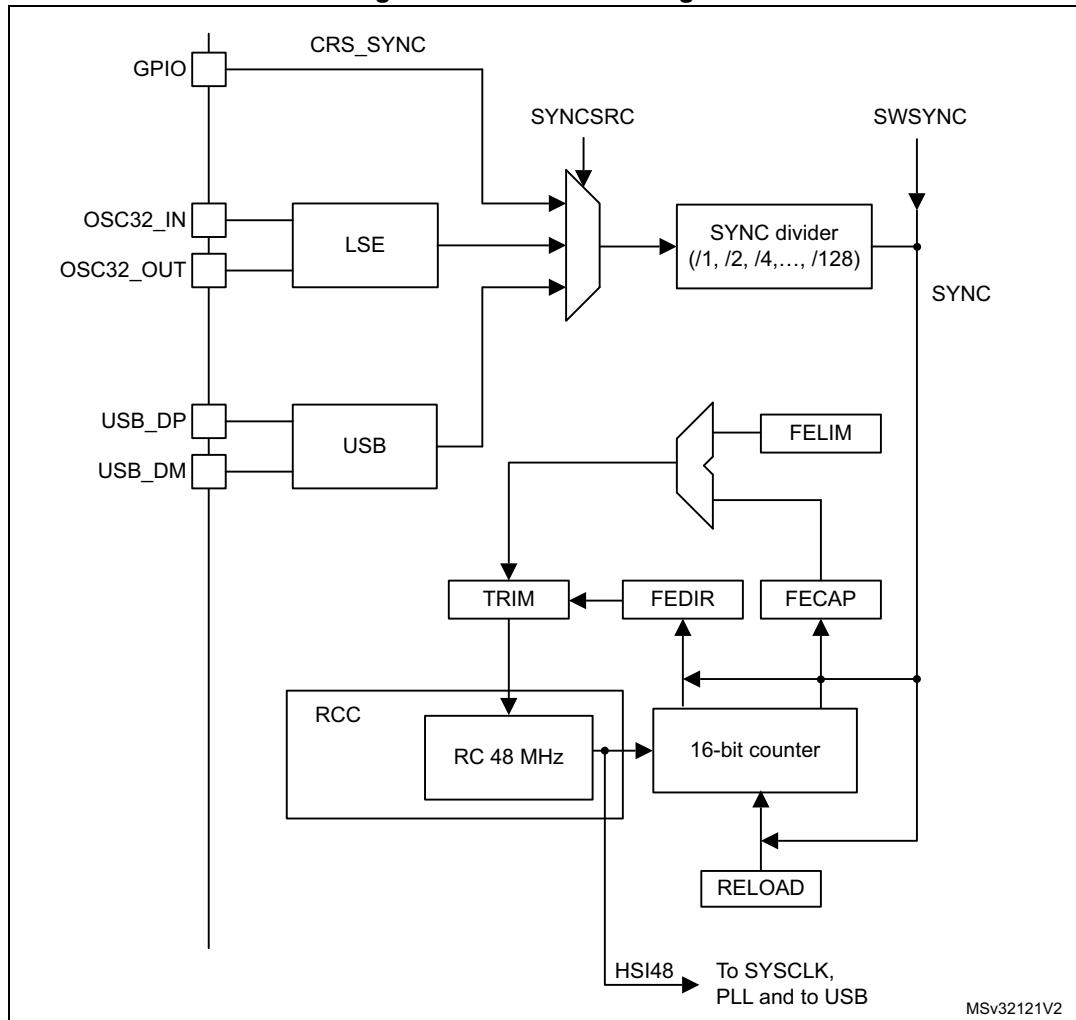
Table 252. CRS features

Feature	CRS1
TRIM width	6 bits

## 40.4 CRS functional description

### 40.4.1 CRS block diagram

Figure 410. CRS block diagram



### 40.4.2 Synchronization input

For more information on the CRS synchronization source configuration, refer to [Section 40.7.2: CRS configuration register \(CRS\\_CFGR\)](#).

It is also possible to generate a synchronization event by software, by setting the **SWSYNC** bit in the **CRS\_CR** register.

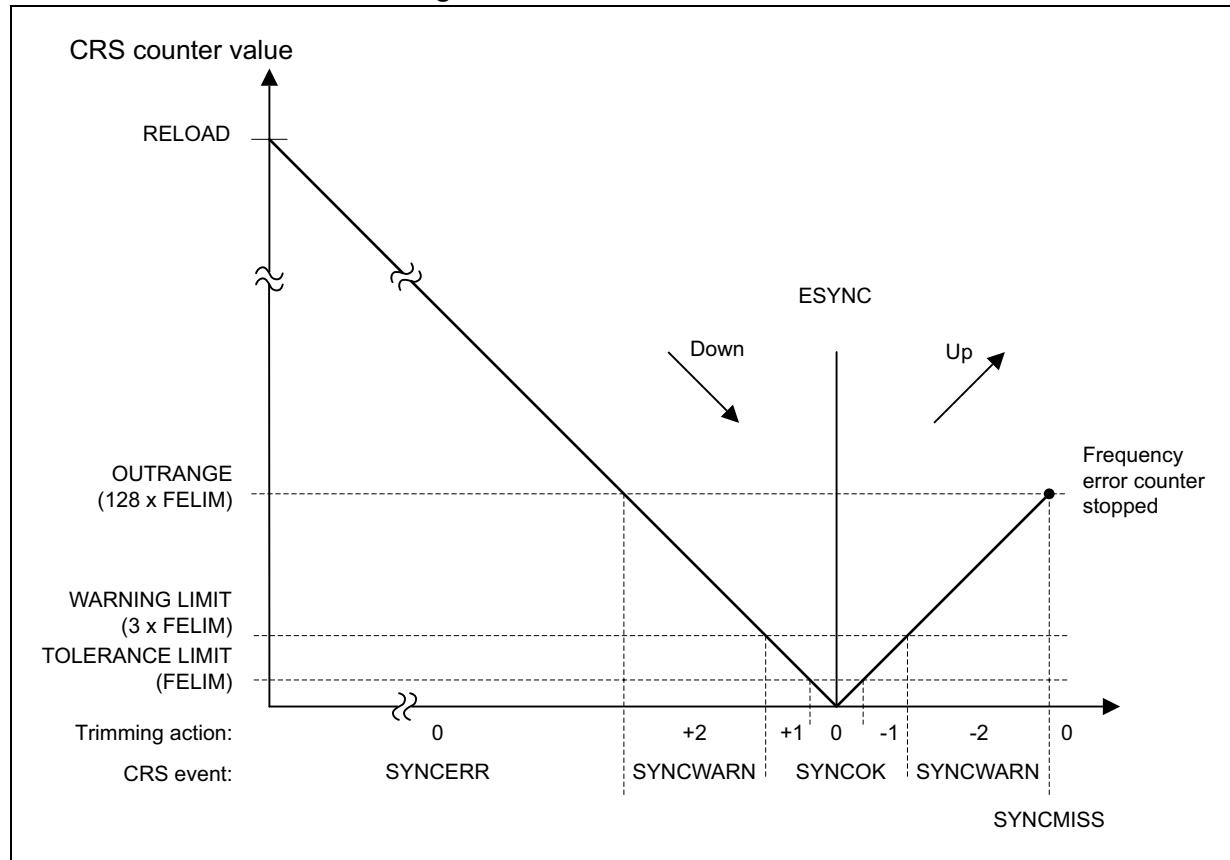
### 40.4.3 Frequency error measurement

The frequency error counter is a 16-bit down/up counter which is reloaded with the **RELOAD** value on each **SYNC** event. It starts counting down till it reaches the zero value, where the **ESYNC** (expected synchronization) event is generated. Then it starts counting up to the **OUTRANGE** limit where it eventually stops (if no **SYNC** event is received) and generates a

SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS\_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS\_ISR register. When the SYNC event is detected during the downcounting phase (before reaching the zero value), it means that the actual frequency is lower than the target (and so, that the TRIM value must be incremented), while when it is detected during the upcounting phase it means that the actual frequency is higher (and that the TRIM value must be decremented).

Figure 411. CRS counter behavior



#### 40.4.4 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS\_CFGR register
- WARNING LIMIT, defined as  $3 \times$  FELIM value
- OUTRANGE (error limit), defined as  $128 \times$  FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming which is enabled by setting the AUTOTRIMEN bit in the CRS\_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.

- SYNCOK status indicated
- TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated
  - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, it means that a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
  - SYNCWARN status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, it means that the frequency is out of the trimming range. This can also happen when the SYNC input is not clean or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

**Note:** *If the actual value of the TRIM field is so close to its limits that the automatic trimming would force it to overflow or underflow, then the TRIM value is set just to the limit and the TRIMOVF status is indicated.*

*In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS\_CR register), the TRIM field of CRS\_CR is adjusted by hardware and is read-only.*

#### 40.4.5 CRS initialization and configuration

##### RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. It is then decreased by one to reach the expected synchronization on the zero value. The formula is the following:

$$\text{RELOAD} = (f_{\text{TARGET}} / f_{\text{SYNC}}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

##### FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of HSI48 oscillator clock ticks. The following formula can be used:

$$\text{FELIM} = (f_{\text{TARGET}} / f_{\text{SYNC}}) * \text{STEP}[\%] / 100\% / 2$$

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to  $(f_{\text{TARGET}} / f_{\text{SYNC}}) = 48000$  and to a typical trimming step size of 0.14%.

**Note:** *The trimming step size depends upon the product, check the datasheet for accurate setting.*

**Caution:** There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields which can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 \* FELIM value (OUTRANGE limit).

## 40.5 CRS low-power modes

**Table 253. Effect of low-power modes on CRS**

Mode	Description
Sleep	No effect. CRS interrupts cause the device to exit the Sleep mode.
Stop	CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator restarted.
Standby	The CRS peripheral is powered down and must be reinitialized after exiting Standby mode.

## 40.6 CRS interrupts

**Table 254. Interrupt control bits**

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCIE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKIE	SYNOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

## 40.7 CRS registers

Refer to [Section 1.2 on page 61](#) for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

### 40.7.1 CRS control register (CRS\_CR)

Address offset: 0x000

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TRIM[5:0]						SW SYNC	AUTO TRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNC WARNIE	SYNC OKIE
		rw	rw	rw	rw	rw	rt_w1	rw	rw		rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **TRIM[5:0]**: HSI48 oscillator smooth trimming

These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI48 oscillator.

The default value is 32, which corresponds to the middle of the trimming interval. The trimming step is specified in the product datasheet. A higher TRIM value corresponds to a higher output frequency.

When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to [Section 40.4.4](#) for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS\_CFG register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ESYNCIE**: Expected SYNC interrupt enable

0: Expected SYNC (ESYNCF) interrupt disabled

1: Expected SYNC (ESYNCF) interrupt enabled

- Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable  
 0: Synchronization or trimming error (ERRF) interrupt disabled  
 1: Synchronization or trimming error (ERRF) interrupt enabled
- Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable  
 0: SYNC warning (SYNCWARNF) interrupt disabled  
 1: SYNC warning (SYNCWARNF) interrupt enabled
- Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable  
 0: SYNC event OK (SYNCOKF) interrupt disabled  
 1: SYNC event OK (SYNCOKF) interrupt enabled

#### 40.7.2 CRS configuration register (CRS\_CFGR)

This register can be written only when the frequency error counter is disabled (CEN bit is cleared in CRS\_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SYNCPOL	Res.	SYNCSRC[1:0]	Res.	SYNCDIV[2:0]			FELIM[7:0]								
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
RELOAD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SYNCPOL**: SYNC polarity selection

This bit is set and cleared by software to select the input polarity for the SYNC signal source.  
 0: SYNC active on rising edge (default)  
 1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection

These bits are set and cleared by software to select the SYNC signal source.  
 00: GPIO selected as SYNC signal source  
 01: LSE selected as SYNC signal source  
 10: USB SOF selected as SYNC signal source (default).  
 11: Reserved

*Note: When using USB LPM (Link Power Management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the HSI48 oscillator on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE or reference clock on the GPIOs should be used as SYNC signal.*

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.

- 000: SYNC not divided (default)
- 001: SYNC divided by 2
- 010: SYNC divided by 4
- 011: SYNC divided by 8
- 100: SYNC divided by 16
- 101: SYNC divided by 32
- 110: SYNC divided by 64
- 111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS\_ISR register. Refer to [Section 40.4.4](#) for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to [Section 40.4.3](#) for more details about counter behavior.

### 40.7.3 CRS interrupt and status register (CRS\_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FECAP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEDIR	Res.	Res.	Res.	Res.	TRIM OVF	SYNC MISS	SYNC ERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNC WARNF	SYNC OKF
r					r	r	r					r	r	r	r

Bits 31:16 **FECAP[15:0]**: Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event.

Refer to [Section 40.4.4](#) for more details about FECAP usage.

Bit 15 **FEDIR**: Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.

0: Upcounting direction, the actual frequency is above the target.

1: Downcounting direction, the actual frequency is below the target.

## Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **TRIMOVF**: Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS\_CR register. It is cleared by software by setting the ERRC bit in the CRS\_ICR register.

0: No trimming error signalized

1: Trimming error signalized

Bit 9 **SYNCMISS**: SYNC missed

This flag is set by hardware when the frequency error counter reached value  $FELIM * 128$  and no SYNC was detected, meaning either that a SYNC pulse was missed or that the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, and that some other action has to be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC) and an interrupt is generated if the ERRIE bit is set in the CRS\_CR register. It is cleared by software by setting the ERRC bit in the CRS\_ICR register.

- 0: No SYNC missed error signalized
- 1: SYNC missed error signalized

Bit 8 **SYNCERR**: SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to  $FELIM * 128$ . This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS\_CR register. It is cleared by software by setting the ERRC bit in the CRS\_ICR register.

- 0: No SYNC error signalized
- 1: SYNC error signalized

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCF**: Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS\_CR register. It is cleared by software by setting the ESYNCC bit in the CRS\_ICR register.

- 0: No expected SYNC signalized
- 1: Expected SYNC signalized

Bit 2 **ERRF**: Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS\_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS\_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

- 0: No synchronization or trimming error signalized
- 1: Synchronization or trimming error signalized

Bit 1 **SYNCWARNF**: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to  $FELIM * 3$ , but smaller than  $FELIM * 128$ . This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS\_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS\_ICR register.

- 0: No SYNC warning signalized
- 1: SYNC warning signalized

Bit 0 **SYNCOKF**: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than  $FELIM * 3$ . This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS\_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS\_ICR register.

- 0: No SYNC event OK signalized
- 1: SYNC event OK signalized

#### 40.7.4 CRS interrupt flag clear register (CRS\_ICR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ESYNCC	ERRC	SYNC WARNC	SYNC OKC											
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCC**: Expected SYNC clear flag

Writing 1 to this bit clears the ESYNCF flag in the CRS\_ISR register.

Bit 2 **ERRC**: Error clear flag

Writing 1 to this bit clears TRIMOVF, SYNCMISS and SYNCERR bits and consequently also the ERRF flag in the CRS\_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag

Writing 1 to this bit clears the SYNCWARNF flag in the CRS\_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag

Writing 1 to this bit clears the SYNCOKF flag in the CRS\_ISR register.

#### 40.7.5 CRS register map

Table 255. CRS register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0x00	CRS_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																	
0x04	CRS_CFGR	SYNPOL	Res.	SYNC SRC [1:0]	Res.	SYNC DIV [2:0]	FELIM[7:0]				RELOAD[15:0]							
	Reset value	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1	1	
0x08	CRS_ISR	FECAP[15:0]										FEDIR	Res.	Res.	Res.	Res.	Res.	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 255. CRS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	CRS_ICR	Res.																															
	Reset value																																

Refer to [Section 2.2 on page 66](#) for the register boundary addresses.

## 41 Debug support (DBG)

### 41.1 Introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Independent breakpoint debugging of each CPU core in the system
- Code execution tracing
- Software instrumentation
- Cross-triggering
- The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The debug features are based on Arm® CoreSight™ components.

- General features:
  - SWJ-DP: JTAG/Serial-wire debug port
  - AHB-AP: AHB access port
- CPU2 debug features:
  - ROM tables
  - System control space (SCS)
  - Breakpoint unit (BPU)
  - Data watchpoint and Trace unit (DWT)
  - Cross trigger interface (CTI)
- CPU1 debug features:
  - ROM table
  - System control space (SCS)
  - Breakpoint unit (FPB)
  - Data watchpoint and Trace unit (DWT)
  - Instrumentation trace macrocell (ITM)
  - Embedded trace macrocell (ETM), only available on STM32WB55xx devices
  - Cross trigger interface (CTI)
  - Trace port interface unit (TPU)

CPU2 debug access via CPU2 AHB-AP and its associated AHB bus is disabled.

The CPU1 debug features are accessible by the debugger via the CPU1 AHB-AP.

Additional information can be found in the Arm® documents referenced in [Section 41.20](#).

### 41.2 Debug use cases

The trace and debug system is designed to support a variety of typical use cases:

- Low cost trace
  - Limited trace capability is available over the single-wire debug output. This supports code instrumentation using “printf”, tracing of data and address watchpoints, interrupt

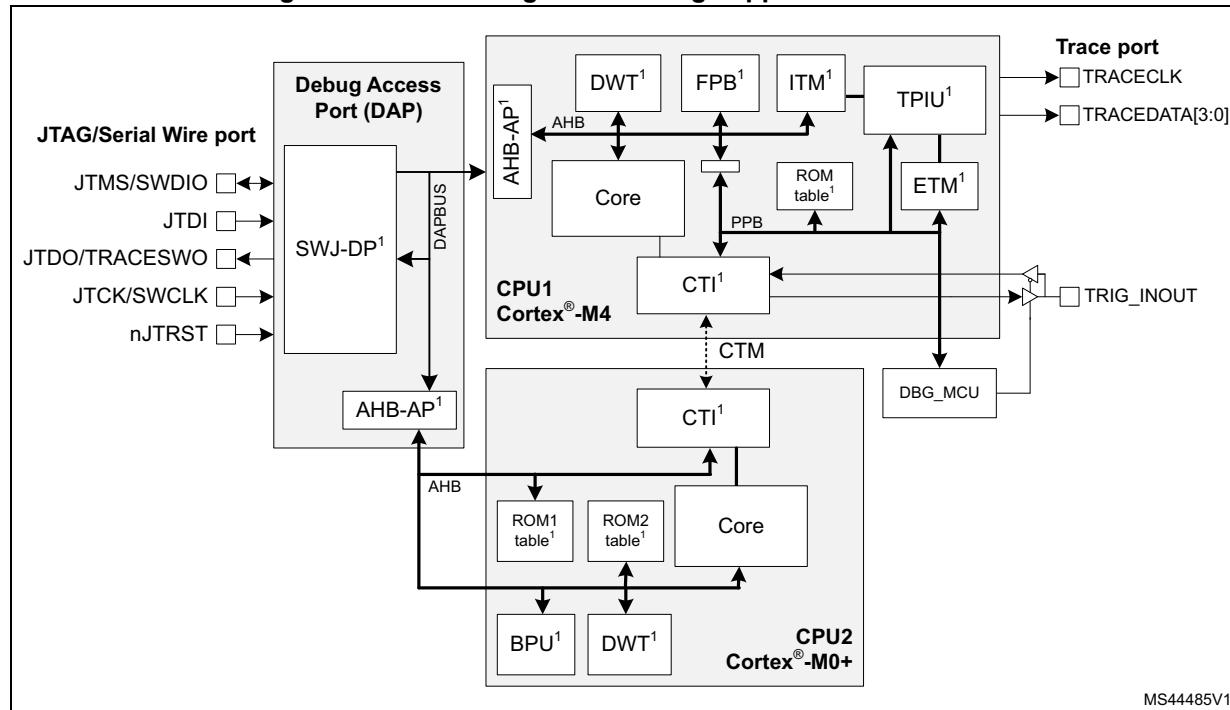
detection and program counter sampling. Single-wire trace can be maintained even when one or both processors are switched off or clock-stopped.

- Breakpoint debugging of each core independently  
Both processor cores can be simultaneously and independently debugged using equipment connected to the JTAG/SWD debug port. This enables, among others, breakpoint and watchpoint setting, code stepping and memory access.
- Synchronous debugging of both cores  
When one core stops due to a breakpoint or a debugger stop command, the other core can be stopped as well. Similarly, the cores can be restarted at the same time. This allows the user to debug loosely coupled applications, which require the processors to remain synchronized.
- Tracing code execution via the trace port  
Trace information from the CPU1 (Cortex®-M4) is combined into a single trace stream and sent to a trace port analyzer in real time. An ID embedded in the trace allows the analyzer to identify the source of each information packet.

## 41.3 DBG functional description

### 41.3.1 DBG block diagram

Figure 412. Block diagram of debug support infrastructure



1. Arm® CoreSight™ component

### 41.3.2 DBG pins and internal signals

Table 256. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	IO	Serial wire data in/out	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	-	-	PB3
nJTRST	I	JTAG test reset	-	-	PB4

**Table 257. Trace port pins**

Pin name	Type	Description	Pin assignment
TRACED0	O	Trace synchronous data out 0	Refer to datasheet
TRACED1		Trace synchronous data out 1	
TRACED2		Trace synchronous data out 2	
TRACED3		Trace synchronous data out 3	
TRACECK		Trace clock	

**Table 258. Single Wire Trace port pins**

Pin name	Type	Description	Pin assignment
TRACESWO	O	Single wire trace asynchronous data out	PB3 <sup>(1)</sup>

1. TRACESWO is multiplexed with JTDO. This means that single wire trace is only available when using the serial wire debug interface, and not when using JTAG.

**Table 259. Trigger pins**

Pin name	Type	Description	Pin assignment
TRIG_INOUT	IO	External trigger bi-directional <sup>(1)</sup>	Refer to datasheet

1. TRIG\_INOUT can be configured as an input or an output by the TRGOEN bit in the DBGMCU.

### 41.3.3 DBG power domains

The debug components are located in the core power domain. This means that debugger connection is not possible in shutdown or standby low power modes. To avoid losing the connection when the device enters standby mode, it is possible to maintain the power to the core by setting a bit in the DBGMCU. This keeps the processor clocks active, and holds off the reset, so that the debug session is maintained.

### 41.3.4 DBG clocks

The debugger supplies the clock for the debug port via the debug interface pin, JTCK/SWCLK. This clock is used to register the serial input data in both serial wire and JTAG mode, as well as to operate the state machines and internal logic of the debug port. It must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DAPCLK domain, which covers the rest of the SWJ-DP and the CPU2 access port.

The DAPCLK is a gated version of the system HCLK4.

The DAPCLK domain is enabled by the debugger using the CDBGWRUPREQ bit in the debug port CTRL/STAT register. The clock must be enabled before the debugger can access any of the debug features on the device. The availability of the clock is reflected in the CDBGWRUPACK bit in the debug port CTRL/STAT register. The DAPCLK is disabled at power up, after OBL, and after wakeup from Standby, and must be disabled when the debugger is disconnected, to reduce power consumption.

The debug and trace components included in the processors (among them ETM, ITM, DWG, FPB) are clocked with the corresponding core clock.

#### 41.3.5 Debug and low power modes

The devices include power saving features that allow the core power domain to be switched off or stopped when not required. If the power is switched off, or the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power saving mode emulation has been implemented. If emulation is enabled for a domain, the domain still enters power saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power saving mode, but the debugger does not lose the connection.

Emulation mode is programmed in the Microcontroller debug (DBGMCU) unit. For more information refer to [Section 41.8](#).

#### 41.3.6 DBG reset

The debug port (SWJ-DP) is reset by a power-on reset or an OBL reset, and when waking up from Standby mode.

### 41.4 Serial wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a Coresight™ component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:

- a 5-pin standard JTAG interface (JTAG-DP)
- a 2-pin (clock + data) “serial-wire debug” port (SW-DP)

The two modes are mutually exclusive, since they share the same IO pins.

By default the JTAG-DP is selected after a system or a power-on reset. The five IO pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO, and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

JTCK/SWCLK must be cycled for each data bit.

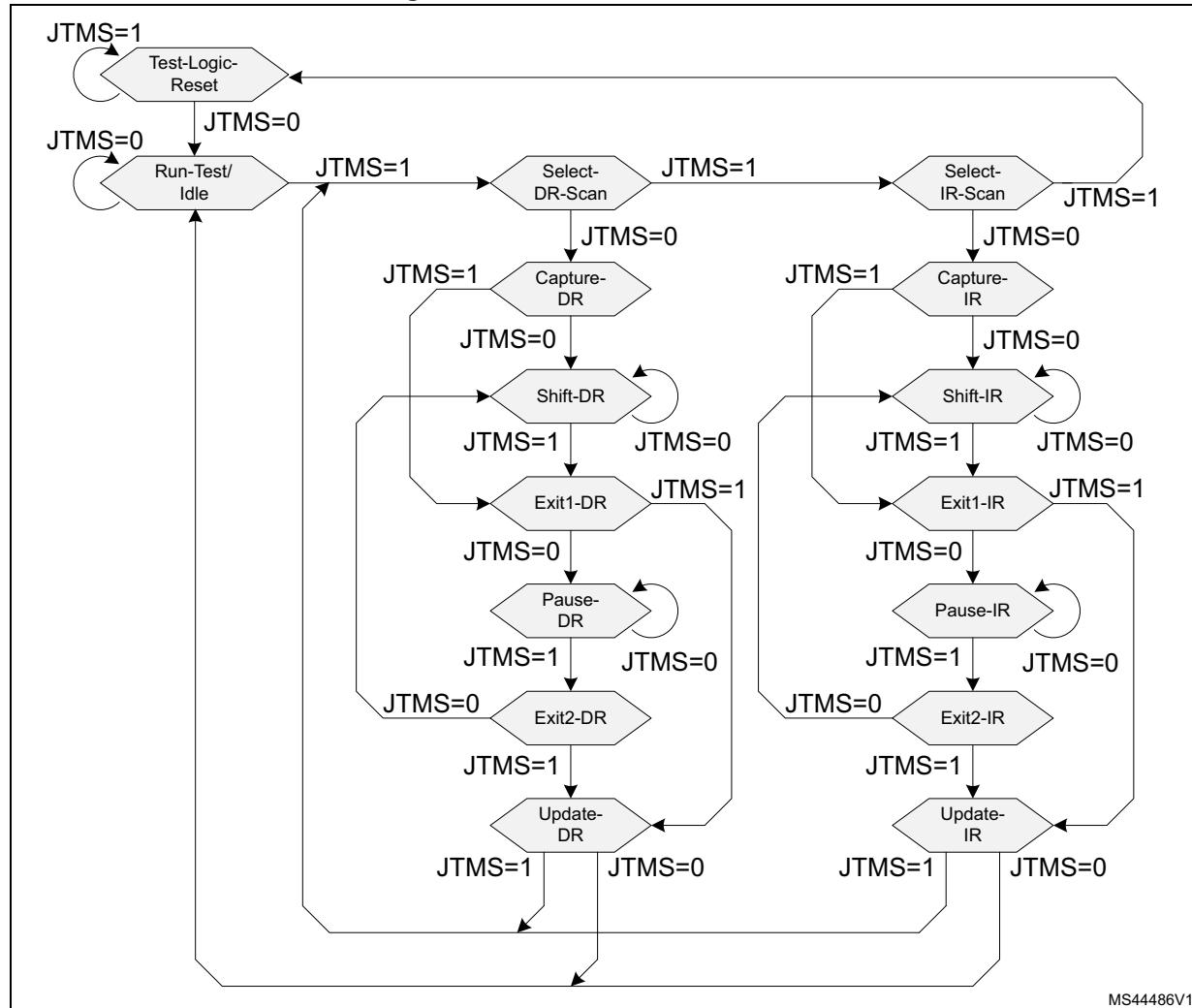
In SW-DP mode, the unused JTAG pins JTDI, JTDO and nJTRST can be used for other functions. It should be noted that all SWJ port IOs can be reconfigured to other functions by software, but debugging is no longer possible.

#### 41.4.1 JTAG debug port

There are two TAPs on the JTAG debug port, the JTAG-DP TAP and the BSC TAP.

The JTAG-DP implements a TAP state machine (TAPSM), shown in [Figure 413](#), based on IEEE Std 1149.1-1990. The state machine controls two scan chains, one associated with an instruction register (IR), and the other one with a number of data registers (DR).

**Figure 413. JTAG TAP state machine**



The operation of the JTAG-DP is as follows:

- When the TAPSM goes through the Capture-IR state, 0b0001 is transferred onto the Instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.
- While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:
  - The LSB of the IR scan chain is output on JTDO.
  - Bit[n] of the IR scan chain is transferred to bit[n-1].
  - The value on JTDI is transferred to the MSB of the IR scan chain.
- When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred into the Instruction register.
- When the TAPSM goes through the Capture-DR state, a value is transferred from one

of the Data registers onto one of the DR scan chains, connected between JTDI and JTDO.

- The value held in the Instruction register determines which Data register (and associated DR scan chain) is selected.
- This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shift in the Shift-IR state.
- When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred into the selected Data register.
- When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the Test-Logic-Reset state.

The data registers corresponding to the 4-bit IR instructions are listed in [Table 260](#). The total IR instruction length is 9 bits.

**Table 260. JTAG-DP data registers**

IR instruction	DR register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	<i>Not implemented</i> : BYPASS selected
1000	ABORT	35	<p><i>ABORT register</i></p> <ul style="list-style-type: none"> <li>– Bits 31:1 = Reserved</li> <li>– Bit 0 = APABORT: write 1 to generate an AP abort.</li> </ul>
1001	(BYPASS)	1	<i>Reserved</i> : BYPASS selected
1010	DPACC	35	<p><i>Debug port access register</i></p> <p>Initiates the debug port and gives access to a debug port register.</p> <ul style="list-style-type: none"> <li>– When transferring data IN:           <ul style="list-style-type: none"> <li>Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request</li> <li>Bits 2:1 = A[3:2] = 2-bit address of a debug port register.</li> <li>Bit 0 = RnW = Read request (1) or write request (0).</li> </ul> </li> <li>– When transferring data OUT:           <ul style="list-style-type: none"> <li>Bits 34:3 = DATA[31:0] = 32-bit data read following a read request</li> <li>Bits 2:0 = ACK[2:0] = 3-bit Acknowledge:               <ul style="list-style-type: none"> <li>010 = OK/FAULT</li> <li>001 = WAIT</li> <li>OTHER = reserved</li> </ul> </li> </ul> </li> </ul>
1011	APACC	35	<p><i>Access port access register</i></p> <p>Initiates an access port and gives access to an access port register.</p> <ul style="list-style-type: none"> <li>– When transferring data IN:           <ul style="list-style-type: none"> <li>Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request</li> <li>Bits 2:1 = A[3:2] = 2-bit sub-address of an access port register.</li> <li>Bit 0 = RnW= Read request (1) or write request (0).</li> </ul> </li> <li>– When transferring data OUT:           <ul style="list-style-type: none"> <li>Bits 34:3 = DATA[31:0] = 32-bit data read following a read request</li> <li>Bits 2:0 = ACK[2:0] = 3-bit Acknowledge:               <ul style="list-style-type: none"> <li>010 = OK/FAULT</li> <li>001 = WAIT</li> <li>OTHER = reserved</li> </ul> </li> </ul> </li> </ul>
1100	(BYPASS)	1	<i>Reserved</i> : BYPASS selected

Table 260. JTAG-DP data registers (continued)

IR instruction	DR register	Scan chain length	Description
1101	(BYPASS)	1	<i>Reserved: BYPASS selected</i>
1110	IDCODE	32	<i>ID Code</i> 0x6BA0 0477: Arm® JTAG debug port ID code
1111	BYPASS	1	<i>Bypass</i> A single JTCK cycle delay is inserted between JTDI and JTDO

The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification [\[1\]](#).

#### 41.4.2 SW debug port

The Serial Wire Debug protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data (100 kΩ pull-up required)

Serial data is transferred LSB first, synchronously with the clock. A transfer comprises three phases:

1. packet request (8 bits) transmitted by the host, see [Table 261](#).
2. acknowledge response (3 bits) transmitted by the target, see [Table 262](#).
3. data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read), see [Table 263](#).

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock cycle turn-around time is inserted.

Table 261. Packet request

Bit field	Name	Description
0	Start	Must be “1”
1	APnDP	– 0: DP register access - see <a href="#">Table 260</a> for a list of DP registers – 1: AP register access - see <a href="#">Section 41.5: Access ports</a>
2	RnW	– 0: Write request – 1: Read request
4:3	A(3:2)	Address field of the DP or AP register (refer to <a href="#">Table 260</a> )
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host, must be read as “1” by target

**Table 262. ACK response**

Bit field	Name	Description
2:0	ACK	<ul style="list-style-type: none"> <li>– 000: FAULT</li> <li>– 010: WAIT</li> <li>– 100: OK</li> </ul>

**Table 263. Data transfer**

Bit field	Name	Description
31:0	WDATA or RDATA	Write or Read data
32	Parity	Single bit parity of 32 data bits

In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled: in this case the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists in 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the Serial Wire debug protocol, refer to the Arm® Debug Interface Architecture Specification [\[1\]](#).

*Note:* The SWJ-DP implements SWD protocol version 2.

#### 41.4.3 Debug port registers

Both the SW-DP and the JTAG-DP access the debug port (DP) registers listed in [Table 264](#).

The debugger can access the DP registers as follows:

- Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a Read or Write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the Packet Request word sent to the SW-DP with the APnDP bit reset (see [Table 261](#)). The write data are sent in the data phase.
- To access one of the banked DP registers at address 0x4, the register number must first be written to the DP\_SELECT register at address 0x8. Any subsequent read or write to address 0x4 accesses the register corresponding to the content of the DP\_SELECT register.

#### 41.4.4 DP debug port identification register (DP\_DPIDR)

Address offset: 0x0

Reset value: 0x5BA0 2477

Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]											
r	r	r	r	r	r	r	r	r	r	r	r	Res.	Res.	Res.	MIN
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[3:0]				DESIGNER[10:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	Res.

Bits 31:28 **REVISION[3:0]**: Revision code

0x5

Bits 27:20 **PARTNO[7:0]**: Part number for the debug port

0xBA

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN**: Minimal debug port (MINDP) implementation

0x0: MINDP not implemented (transaction counter and pushed operations are supported)

Bits 15:12 **VERSION[3:0]**: DP architecture version

0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code

0x23B: Arm® JEDEC code

Bit 0 Reserved, must be kept at reset value.

#### 41.4.5 DP abort register (DP\_ABORTR)

Address offset: 0x0

Reset value: 0x0000 0000

Write only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ORUNERR CLR	WDERR CLR	STKERR CLR	STKCMP CLR	DAPABORT										
											w	r	r	r	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: Overrun error clear

0: No effect

1: Clear CTRL/STAT.STICKYORUN bit

Bit 3 **WDERRCLR**: Write data error clear

0: No effect

1: Clear CTRL/STAT.WDATAERR bit

Bit 2 **STKERRCLR**: Sticky error clear

0: No effect

1: Clear CTRL/STAT.STICKYERR bit

Bit 1 **STKCMPCCLR**: Sticky compare clear

0: No effect

1: Clear CTRL/STAT.STICKYCMP bit

Bit 0 **DAPABORT**: Aborts current AP transaction if an excessive number of WAIT responses are returned, indicating that the transaction is stalled.

0: No effect

1: Abort transaction

#### 41.4.6 DP control and status register (DP\_CTRL/STATR)

Address offset: 0x4 and DP\_SELECTR.DPBANKSEL = 0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CDBGWRUPACK	CDBGWRUPREQ	Res.	Res.	Res.	Res.	TRNCNT[11:4]							
		r	r					r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRNCNT[3:0]				CMASKLANE[3:0]				WDATAERR	READOK	STICKYERR	STICKYCMP	TRNMODE[1:0]		STICKYORUN	ORUNDETECT
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CDBGWRUPACK**: See description in [Section 41.3.4: DBG clocks](#).

0 = DAPCLK gated

1 = DAPCLK enabled

Bit 28 **CDBGWRUPREQ**: Controls the DAPCLK enable request signal.

0 = Requests DAPCLK gating

1 = Requests DAPCLK enable

Bits 27:24 Reserved, must be kept at reset value.

- Bits 23:12 **TRNCNT[11:0]**: Transaction counter. To program a sequence of transactions to incremental addresses via an AP, TRNCNT is loaded with the number of transactions to perform. It is decremented at the successful completion of each transaction.
- Bits 11:8 **MASKLANE[3:0]**: Indicates the bytes to be masked in pushed-compare and pushed-verify operations (CTRL/STAT.TRNMODE = 1 or 2). In the pushed operations, the word supplied in an AP write transaction is compared with the current value at the target AP address.
- 0b1XXX = include byte lane 3 in comparisons
  - 0bX1XX = include byte lane 2 in comparisons
  - 0bXX1X = include byte lane 1 in comparisons
  - 0bXXX1 = include byte lane 0 in comparisons
- Bit 7 **WDATAERR**. Write data error (read only) in SW-DP. Indicates that:
- there is a parity or framing error on the data phase of a write, or
  - a write that has been accepted by the DP is then discarded without being submitted to the AP.
- This bit is reset by writing 1 to the ABORT.WDERRCLR bit.
- 0: No error
  - 1: Error has occurred
- Reserved in JTAG-DP.
- Bit 6 **READOK**. AP read response (read only) in SW-DP. Indicates the response to the last AP read access.
- 0: Read not OK
  - 1: Read OK
- Reserved in JTAG-DP.
- Bit 5 **STICKYERR**. Transaction error (read only in SW-DP, R/W in JTAG-DP). Indicates that an error occurred in an AP transaction.
- 0: No error
  - 1: Error has occurred
- In the SW-DP, this bit is reset by writing 1 to the ABORT.STKERRCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.
- Bit 4 **STICKYCMP**. Compares match (read only in SW-DP, R/W in JTAG-DP). Indicates that a match occurred in a pushed operation.
- 0: Match if TRNMODE = 0x1; no match if TRNMODE = 0x2
  - 1: No match if TRNMODE = 0x1; match if TRNMODE = 0x2
- In the SW-DP, this bit is reset by writing 1 to the ABORT.STKCMPCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.
- Bits 3:2 **TRNMODE[1:0]**: Transfer mode for AP write operations (for read operations, this field must be set to 0x0).
- 0x0: Normal operation. AP transactions are passed directly to the AP.
  - 0x1: Pushed-verify operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data and if they do not match, the STICKYCMP bit is set.
  - 0x2: Pushed-compare operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data and if they match, the STICKYCMP bit is set.
  - 0x3: reserved
- In pushed operation, only the data bytes indicated by the MASKLANE field are included in the compare.

Bit 1 **STICKYORUN**. Overrun (read only in SW-DP, R/W in JTAG-DP). Indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.

0: No overrun

1: Overrun occurred

In the SW-DP, this bit is reset by writing 1 to the ABORT.ORUNERRRCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bit 0 **ORUNDETECT**. Overrun detection mode enable.

0: Overrun detection disabled

1: Overrun detection enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

#### 41.4.7 DP data link control register (DP\_DLCR)

Address offset: 0x4 and DP\_SELECTR.DPBANKSEL = 1

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]	Res.								
						r	r								

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]**: Tristate period for SWDIO.

0x0: 1 data bit period

0x1: 2 data bit periods

0x2: 3 data bit periods

0x3: 4 data bit periods

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value (set to 1).

Bits 5:0 Reserved, must be kept at reset value.

#### 41.4.8 DP target identification register (DP\_TARGETIDR)

Address offset: 0x4 and DP\_SELECTR.DPBANKSEL = 2

Reset value: 0x0495 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TREVISION[3:0]				TPARTNO[15:4]											
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
TPARTNO[3:0]				TDESIGNER[10:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **TREVISION**: Target revision.

0x0: revision 1

Bits 27:12 **TPARTNO**: Target part number.

0x4950: STM32WB55xx/35xx

Bits 11:1 **TDESIGNER**: Target designer JEDEC code.

0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value (set to 1)

#### 41.4.9 DP data link protocol identification register (DP\_DLPIDR)

Address offset: 0x4 and DP\_SELECTR.DPBANKSEL = 3

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				Res.											
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]
												r	r	r	r

Bits 31:28 **TINSTANCE[3:0]**: Target instance number. Defines the instance number for this device in a multi-drop system.

0x0: Instance number 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial Wire Debug protocol version.

0x1: Version 2

#### 41.4.10 DP resend register (DP\_RESENDR)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESEND[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESEND[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RESEND**: Returns the value that was returned by the last AP read or DP RDBUFF read. Used in the event of a corrupted read transfer.

#### 41.4.11 DP access port select register (DP\_SELECTR)

Address offset: 0x8

Reset value: Unknown

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[3:0]															
w	w	w	w												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:28 **APSEL[3:0]**: Access port select. Selects the access port for the next transaction.

0x0: AP0 - CPU1 (Cortex<sup>®</sup>-M4) debug access port (AHB-AP)

0x1: AP1 - CPU2 (Cortex<sup>®</sup>-M0+) debug access port (AHB-AP)

0x2 to 0xF: reserved

Bits 27:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select. Selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select. Selects the register at address 0x4 of the debug port.

0x0: CTRL/STAT register

0x1: DLCR register

0x2: TARGETID register

0x3: DLPIDR register

0x4 to 0xF: Reserved

#### 41.4.12 DP read buffer register (DP\_BUFFR)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDBUFF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDBUFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDBUFF[31:0]**: Contains the value that was returned by the last AP read access. The value returned by an AP read access can either be obtained using a second read access to the same address, which initiates a new transaction on the corresponding bus, or else it can be read from this register, in which case no new AP transaction occurs.

#### 41.4.13 DP target selection register (DP\_TARGETSELR)

Address offset: 0xC

Reset value: Unknown

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
TINSTANCE[3:0]				TPARTNO[15:4]												
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	
TPARTNO[3:0]				TDESIGNER[10:0]												Res.
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w		

Bits 31:28 **TINSTANCE[3:0]**: Target instance number. Defines the instance number for the target device in a multi-drop system. These bits must be written with the same value used for DLPIDR.TINSTANCE to select this device.

Bits 27:12 **TPARTNO[15:0]**: Target part number. Defines the part number for the target device. These bits must be written with the same value used for TARGETID.TPARTNO to select this device.

Bits 11:1 **TDESIGNER[10:0]**: Target designer JEDEC code. Defines the JEDEC code for the target device. These bits must be written with the same value used for TARGETID.TDESIGNER to select this device.

Bit 0 Reserved, must be kept at reset value (set to 1).

#### 41.4.14 Debug port register map and reset values

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interface.

The debug port address is 2-bit wide, defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

**Table 264. Debug port register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0b00	DP_DPIDR	REVISION [3:0]			PARTNO[7:0]			MIN			VERSION [3:0]			DESIGNER[10:0]																					
	Reset value	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	1	1	1	1				
0b00	DP_ABORTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	Reset value																																		
0b01 <sup>(1)</sup>	DP_CTRL/STATR	Res.	Res.	0	CDBGPWRUPACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	Reset value			0	0																														
0b01 <sup>(1)</sup>	DP_DLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	Reset value																																		
0b01 <sup>(2)</sup>	DP_TARGETIDR	TREVISION [3:0]			TPARTNO[15:0]			TDESIGNER[10:0]																											
	Reset value	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0b01 <sup>(3)</sup>	DP_DLPIDR	TINSTANCE [3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	Reset value	0	0	0	0																											0	0	0	1
0b10	DP_RESENRD	RESEND[0:31]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0b10	DP_SELECTR	APSEL[3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	Reset value	x	x	x	x																									x	x	x	x	x	x
0b11	DP_BUFFR	RDBUFF[0:31]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0b11	DP_TERGETSELR	TINSTANCE [3:0]			TPARTNO[15:0]			TDESIGNER[10:0]																											
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		

1. DP\_SELECTR.DPBANKSEL = 1.
2. DP\_SELECTR.DPBANKSEL = 2.
3. DP\_SELECTR.DPBANKSEL = 3.

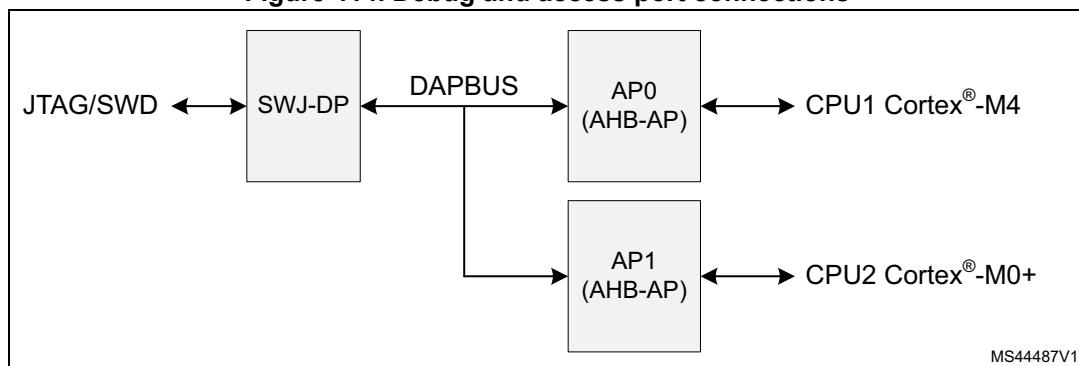
## 41.5 Access ports

As shown in [Figure 414](#), there are two access ports (AP) attached to the DP:

1. AP0: CPU1 (Cortex®-M4) access port (AHB-AP): enables access to the debug and trace features integrated in the Cortex®-M4 processor core via its internal AHB bus.
  2. AP1: CPU2 (Cortex®-M0+) access port (AHB-AP): enables access to the debug and trace features integrated in the Cortex®-M0+ processor core via its internal AHB bus.

Both access ports are of type MEM-AP, the debug and trace component registers are mapped in the address space of the associated debug bus. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in [Table 265](#).

**Figure 414. Debug and access port connections**



The address of the AP registers is composed of

- Bits [7:4]: content of the SELECT register APBANKSEL field in the DP (see [Section 41.4.11](#))
  - Bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see [Table 264](#)) or of the SW-DP Packet Request (see [Table 261](#)), depending on the debug interface used
  - Bits [1:0]: Always set to 0

The content of the SELECT register APSEL field in the DP defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program the SELECT register APSEL field in the DP to choose one of the APs, and the APBANKSEL field to select the register bank to be accessed (see [Section 41.4.11](#)).
  2. Program the A(3:2) field in the APACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a Read or Write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the Packet Request word sent to the SW-DP with the APnDP bit set (see [Table 261](#)). The write data is sent in the data phase.

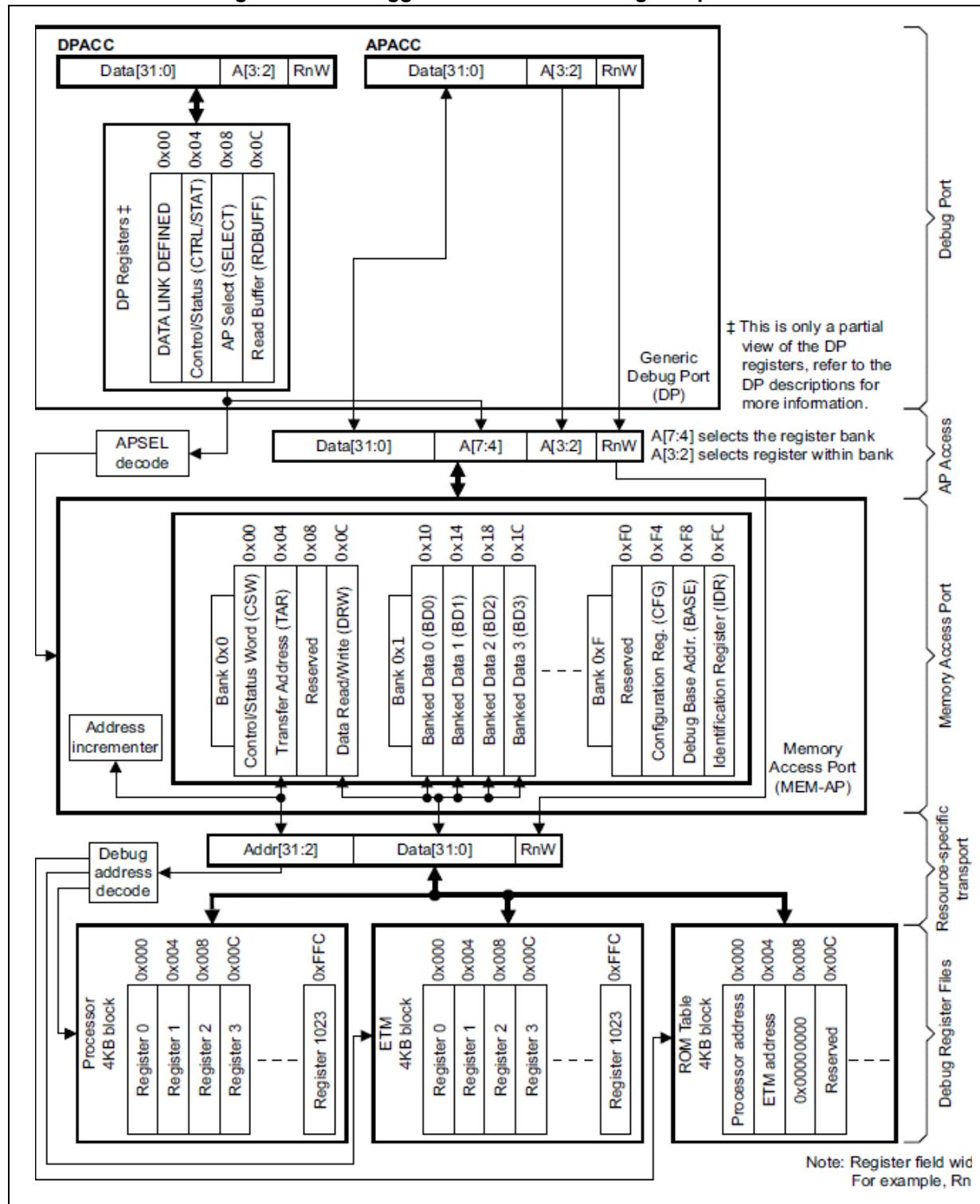
The debugger can access the memory mapped debug component registers through the MEM-AP registers (i.e. using the above AP register access procedure) as follows:

1. Program the transaction target address in the TAR register.
2. Program the CSW register, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the DRW register to initiate a bus transaction at the address held in the TAR register. Alternatively, a read or write to Banked data register BDn triggers an access to address  $\text{TAR}[31:4] + n$  (this enables up to four consecutive addresses to be accessed without changing the address in the TAR register).

*Figure 415* shows how the MEM-AP is used to connect the debug port to the debug components (in this example, a processor, an ETM and a ROM table).

For more detailed information on the MEM-AP refer to the Arm® Debug Interface Architecture Specification [\[1\]](#).

Figure 415. Debugger connection to debug components



### 41.5.1 AP control/status word register (AP\_CSWR)

Address offset: 0x0

Reset value: 0x2300 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	SPROT	Res.	PROT[4:0]						SPI STATUS	Res.	Res.	Res.	Res.	Res.	Res.	
	r		r	r	r	r	r									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	Res.	Res.	Res.	MODE[3:0]						TRIN PROG	DEVICE EN	ADDRIN[1:0]		Res.	SIZE[2:0]	
				r	r	r	r	r	r	r	r	r	r	r	r	

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SPROT**: Secure transfer request. In the AHB-APs this field sets the protection attribute HPROT[6] of the bus transfer.

0: If SPIDEN is high, secure transfer. If SPIDEN is low, non-secure transfer.

1: Non-secure transfer.

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **PROT[4:0]**: Bus transfer protection. In the AHB-APs this field sets the protection attributes HPROT[4:0] of the bus transfer.

0bXXXX0: Instruction fetch

0bXXXX1: Data access

0bXXX0X: User mode

0bXXX1X: Privileged mode

0bXX0XX: Non-bufferable

0bXX1XX: Bufferable

0bX0XXX: Non-cacheable

0bX1XXX: Cacheable

0b0XXXX: Non-exclusive

0b1XXXX: Exclusive

Bit 23 **SPISTATUS**: Status of SPIDEN option bit (read only). This signal determines whether the debugger can access secure memory.

0: Secure AHB transfers are blocked

1: Secure AHB transfers are allowed

Bits 22:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: Barrier support enabled. Defines if memory barrier operation is supported.

0x0: Not supported

Bit 7 **TRINPROG**: Transfer in progress (read only). Indicates if a bus transfer is in progress on the AP.

0x0: No transfer in progress.

0x1: Bus transfer in progress.

Bit 6 **DEVICEEN**: Device enabled (read only). Defines whether the AP can be accessed.

0x1: AP access enabled.

Bits 5:4 **ADDRINC[1:0]**: Auto-increment mode. Defines whether TAR address is automatically incremented after a transaction.

0x0: No auto-increment

0x1: Address is incremented by the size in bytes of the transaction (SIZE field).

0x2: Packed transfers enabled. A 32-bit AP access gives rise to 1 x 32-bit, 2 x 16-bit or 4 x 8-bit bus transactions corresponding to the programmed transaction size. The data is packed or unpacked accordingly.

0x3: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: Size of next memory access transaction

0x0: Byte (8-bit)

0x1: Halfword (16-bit)

0x2: Word (32-bit)

0x3-0x7: reserved

### 41.5.2 AP transfer address register (AP\_TAR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TA[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
TA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TA[31:0]**: Address of current transfer

### 41.5.3 AP data read/write register (AP\_DRWR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TD[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
TD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TD[31:0]**: Data of current transfer

### 41.5.4 AP banked data registers (AP\_BD0-3R)

Address offset: 0x10 (DB0R)

Address offset: 0x14 (BD1R)

Address offset: 0x18 (BD2R)

Address offset: 0x1C (BD3R)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBD[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
TBD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TBD[31:0]**: Banked data of current transfer to address TAR. TA+ AP\_BDnR address [3:2] + 0b00. Auto address incrementing is not performed on AP\_BD0-3R. Banked transfers are only supported for word transfers.

#### 41.5.5 AP base address register (AP\_BASER)

Address offset: 0xF8

Reset value: 0xE00F F003 (AP0), 0xF000 0003 (AP1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[19:4]															
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
BASEADDR[3:0]				Res.	FORMAT	ENTRY PRESENT									
r	r	r	r											r	r

Bits 31:12 **BASEADDR[19:0]**: Base address (bits 31 to 12) of ROM table for the AP. The twelve LSBs are 0 since the ROM table must be aligned on a 4 Kbytes boundary.

AP0 CPU1 (Cortex®-M4) AHB-AP: 0xE00FF

AP1 CPU2 (Cortex®-M0+) AHB-AP: 0xF0000

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: Base address register format.

1: Arm® debug interface v5.

Bit 0 **ENTRYPRESENT**: Indicates that debug components are present on the access port bus.

1: Debug components are present

#### 41.5.6 AP identification register (AP\_IDR)

Address offset: 0xFC

Reset value: 0x2477 0011 (AP0), 0x6477 0001 (AP1)

Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDEC BANK[3:0]				JEDEC CODE[6:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**:

0x2: r0p3  
0x6: r0p7

Bits 27:24 **JEDEC BANK[3:0]**: JEDEC bank.

0x4: Arm®

Bits 23:17 **JEDEC CODE[6:0]**: JEDEC code.

0x3B: Arm®

Bit 16 **MEMAP**: Memory access port.

0x1: Standard register map

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**: Identifies the type of AP.

0x11: CPU1 (Cortex®-M4) AHB-AP (AP0)  
0x01: CPU2 (Cortex®-M0+) AHB-AP (AP1)

### 41.5.7 Access port register map and reset values

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interface.

The access port address is 8-bit wide, defined by debug port register DP\_SELECTR.APBANKSEL[3:0] field and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

**Table 265. Access port register map and reset values**

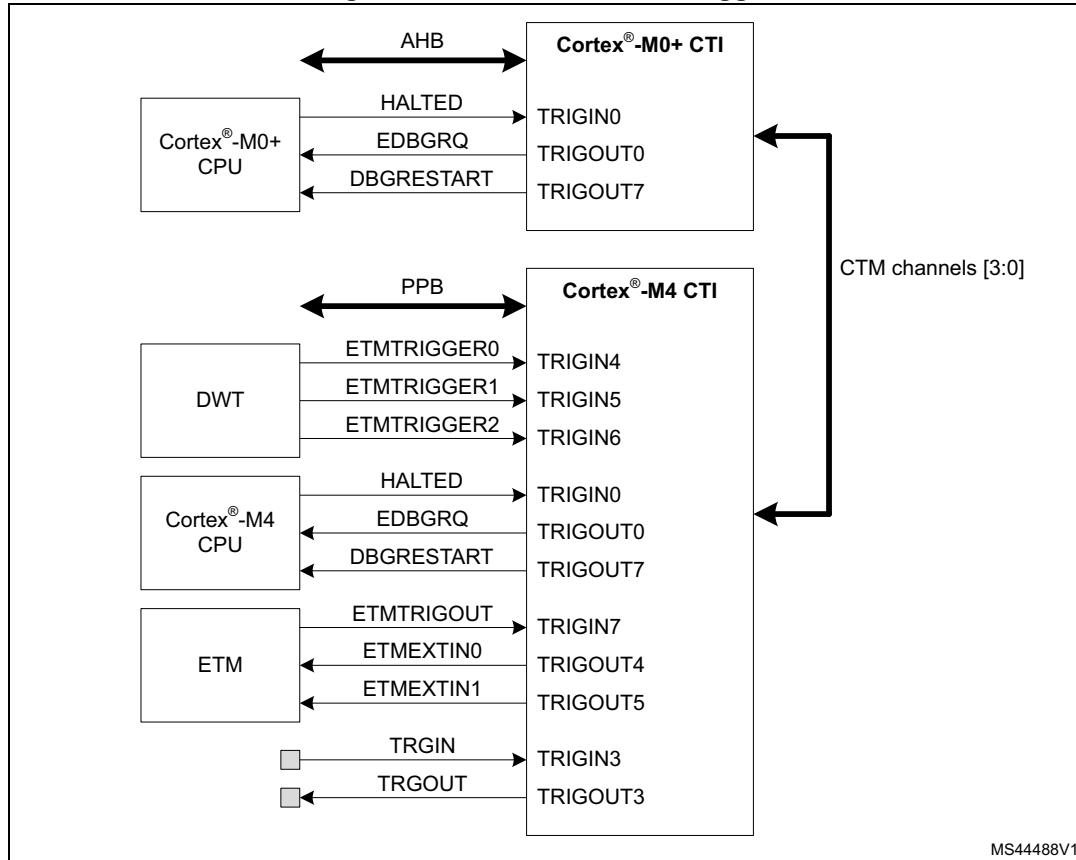
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	AP_CSWR	Res.	SPROT	Res.	PROT[4:0]		SPISTRATUS		Res.	TRINPROG[1:0]	ADDRIN[1:0]	Res.	Res.	Res.	Res.	SIZE[2:0]	0																	
	Reset value	0	0	0	0	1	1	0																0	0	0	0	1	0	0	0	0		
0x04	AP_TAR																								TA[0:31]									
	Reset value	0	0	0	0	0	0	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	AP_DRWR																								TD[0:31]									
	Reset value	0	0	0	0	0	0	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	AP_BD0R																								TBD[0:31]									
	Reset value	0	0	0	0	0	0	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	AP_BD1R																								TBD[0:31]									
	Reset value	0	0	0	0	0	0	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	AP_BD2R																								TBD[0:31]									
	Reset value	0	0	0	0	0	0	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	AP_BD3R																								TBD[0:31]									
	Reset value	0	0	0	0	0	0	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	AP_BASER																								BASEADDR[0:19]									
	Reset value (AP0)	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
	Reset value (AP1)	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1			
0xFC	AP_IDR	REViSION[3:0]	JEDEC BANK[3:0]																						JEDEC CODE[6:0]	MEMMAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]
	Reset value (AP0)	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
	Reset value (AP1)	0	1	1	0	0	1	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			

## 41.6 Cross trigger interface (CTI) and matrix (CTM)

The Cross trigger interfaces (CTIs) and Cross trigger matrix (CTM), taken together, form the CoreSight™ embedded cross trigger (see [Figure 416](#)).

There are two CTI components, one dedicated to the CPU2 and one dedicated to the CPU1. The CTIs are connected to each other via the CTM. The CTI registers are accessible to the debugger via the corresponding access port and associated AHB.

**Figure 416. Embedded cross trigger**



The CTIs enable events from various sources to trigger debug and/or trace activity. For example, a breakpoint reached in one of the processor cores can stop the other processor, or a transition detected on an external trigger input can start code trace.

Each CTI has up to eight trigger inputs and eight trigger outputs. Any input can be connected to any output, on the same CTI, or on another CTI via the CTM.

The trigger input and output signals for each CTI are listed in tables [266](#) to [269](#).

**Table 266. CPU2 CTI inputs**

No.	Source signal	Source component	Comments
0	HALTED	CPU2	CPU2 halted - Indicates CPU2 is in debug mode
1	-	-	Not used
2	-	-	Not used

**Table 266. CPU2 CTI inputs (continued)**

No.	Source signal	Source component	Comments
3	-	-	Not used
4	-	-	Not used
5	-	-	Not used
6	-	-	Not used
7	-	-	Not used

**Table 267. CPU2 CTI outputs**

No.	Output signal	Destination component	Comments
0	EDBGRQ	CPU2	CPU2 halt request - Puts CPU2 in debug mode
1	-	-	Not used
2	-	-	Not used
3	-	-	Not used
4	-	-	Not used
5	-	-	Not used
6	-	-	Not used
7	DBGRESTART	CPU2	CPU2 restart request - CPU2 exits debug mode

**Table 268. CPU1 CTI inputs**

No.	Source signal	Source component	Comments
0	HALTED	CPU1	CPU1 halted - Indicates CPU1 is in debug mode
1	-	-	Not used
2	-	-	Not used
3	-	-	Not used
4	ETMTRIGGER0	CPU1 DWT	Trace trigger - Enables CPU1 execution trace
5	ETMTRIGGER1	CPU1 DWT	Trace trigger - Enables CPU1 execution trace
6	ETMTRIGGER2	CPU1 DWT	Trace trigger - Enables CPU1 execution trace
7	ETMTRIGOUT <sup>(1)</sup>	CPU1 ETM	ETM triggered - Indicates CPU1 trace active

1. Only available on STM32WB55xx devices.

**Table 269. CPU1 CTI outputs**

No.	Source signal	Source component	Comments
0	EDBGRQ	CPU1	CPU1 halt request - Puts CPU1 in debug mode
1	-	-	Not used
2	-	-	Not used

Table 269. CPU1 CTI outputs (continued)

No.	Source signal	Source component	Comments
3	-	-	Not used
4	ETMEXTIN0	CPU1 ETM	ETM trig 0 request - Enables CPU1 execution trace
5	ETMEXTIN1	CPU1 ETM	ETM trig 0 request - Enables CPU1 execution trace
6	-	-	Not used
7	DBGRESTART	CPU1	CPU1 restart request - CPU1 exits debug mode

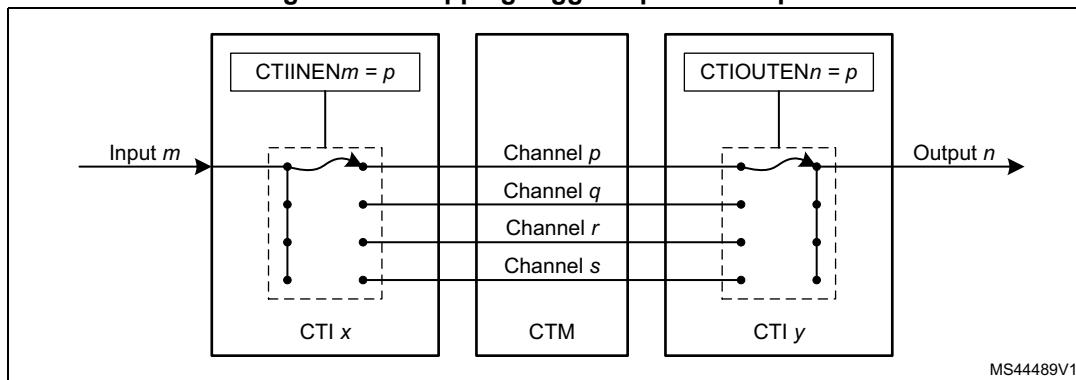
There are four event channels in the cross trigger matrix, thus enabling up to four, parallel, bidirectional connections between trigger inputs and outputs on different CTIs. To connect input number  $m$  on CTI  $x$  to output number  $n$  on CTI  $y$ , the input must be connected to an event channel  $p$  using the  $CTIINENm$  register of CTI  $x$ . The same channel  $p$  must be connected to the output using the  $CTIOUTENn$  register of CTI  $y$ .

*Note:*

*This applies even if the input and output belong to the same CTI.*

An input can be connected to more than one channel (up to four), so an input can be routed to several outputs. Similarly, an output can be connected to several inputs. It is also possible to connect several inputs/outputs to the same channel.

Figure 417. Mapping trigger inputs to outputs



### Example configurations

When either CPU core hits a breakpoint, stop the other core. Restart the two cores synchronously.

To stop both cores when one of them stops the HALTED output of each core must be connected to the EDBGREQ input of the opposite core.

Referring to [Table 266](#) and [Table 268](#), we see that the HALTED signal from the CPU2 is connected to input 0 of the CPU2 CTI, and the same signal from the CPU1 is connected to the same input on the CPU1 CTI. Hence we program the CTIINEN0 register on each CTI to connect these inputs to a CTM channel (eg. channel 0).

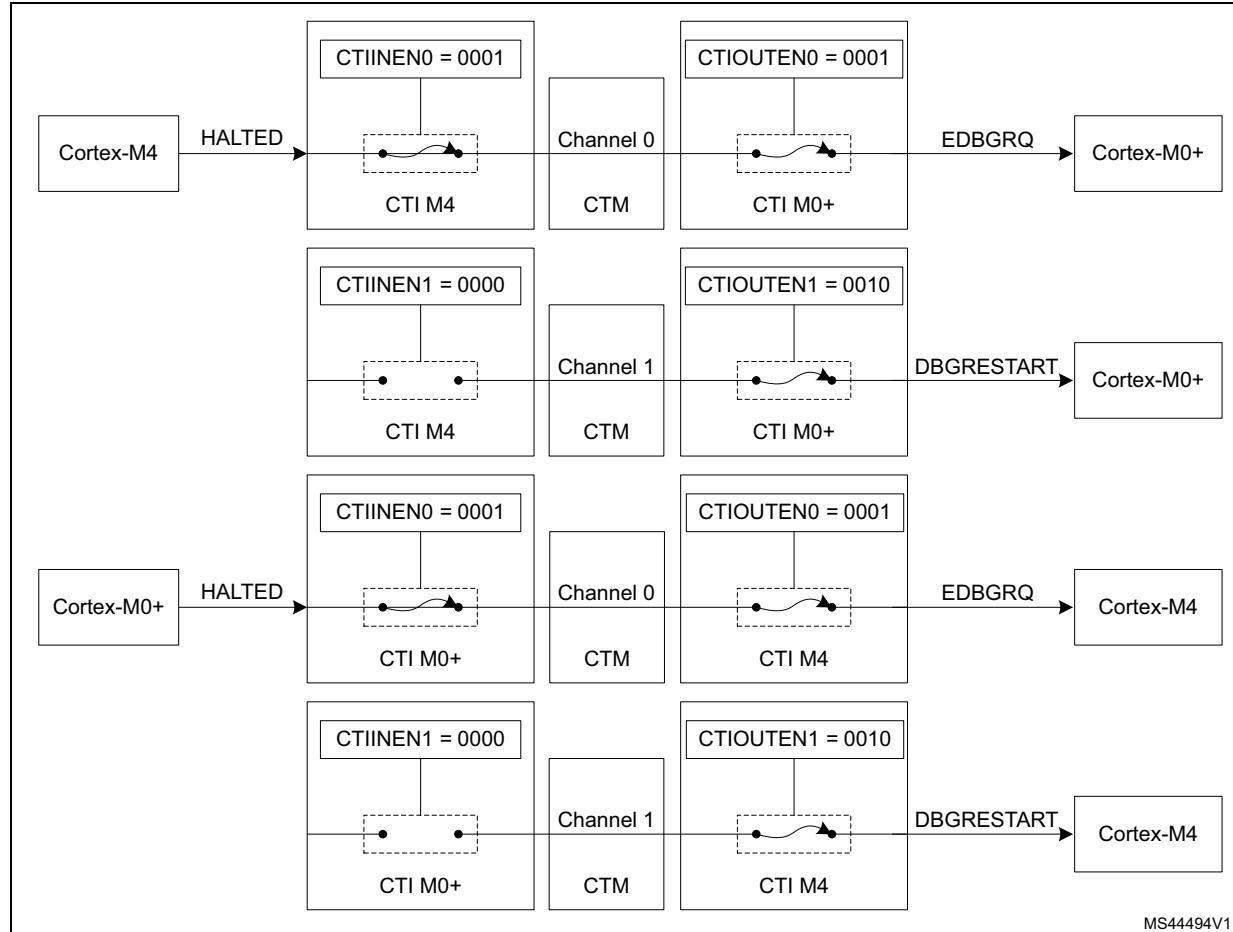
From [Table 267](#) and [Table 269](#) we see that the EDBGREQ signals to the CPUs are connected to output 0 of the respective CTIs. So we program the CTIOUTEN0 register on each CTI to connect these outputs to the same CTM channel.

To restart both cores simultaneously the debugger must use the APPPULSE register in one of the CTIs. This allows the debugger to generate a pulse on any of the four CTM channels. The channel must be connected to the DBGRESTART signal of both cores.

From [Table 267](#) and [Table 269](#) we see that the DBGRESTART signals to the CPUs are connected to output 1 of the respective CTIs. So we program the CTIOUTEN1 register on each CTI to connect these outputs to an unused CTM channel (e.g. channel 1).

The above configuration is illustrated in [Figure 418](#).

**Figure 418. Cross trigger configuration example**



To force the processors to restart simultaneously use the following procedure:

1. Clear the debug request by writing 0x01, then 0x00, to the CTIINTACK register in each CTI.
2. Cause a pulse on channel 1 by writing 0x02 to the APPPULSE register in either CTI. This generates a restart request to both processors.

Note that the debugger can also force both cores to stop simultaneously by writing 0x01 to the APPPULSE register in either CTI, which will generate a pulse on channel 0.

For more information on the Cross-Trigger Interface CoreSight™ component refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual [\[2\]](#).

## 41.7 Cross trigger interface registers

The register file base address is 0xE0043000 for CPU1 CTI and 0xF0001000 for CPU2 CTI. CPU1 CTI and CPU2 CTI are accessed via different access ports. The registers are the same for each CTI.

### 41.7.1 CTI control register (CTI\_CONTROLR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GLBEN														
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN**: Global enable.

- 0: Cross-triggering disabled
- 1: Cross-triggering enabled

### 41.7.2 CTI trigger acknowledge register (CTI\_INTACKR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.							INTACK[7:0]								
									rw						

Bits 31:8 Reserved, must be kept at reset value.

Bit 7:0 **INTACK[7:0]**: Trigger acknowledge.

There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

### 41.7.3 CTI application trigger set register (CTI\_APPSETR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	APPSET[3:0]														
															rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3:0 **APPSET[3:0]**: Set channel event.

Read:

0bXXX0: Channel 0 event inactive  
 0bXXX1: Channel 0 event active  
 0bXX0X: Channel 1 event inactive  
 0bXX1X: Channel 1 event active  
 0bX0XX: Channel 2 event inactive  
 0bX1XX: Channel 2 event active  
 0b0XXX: Channel 3 event inactive  
 0b1XXX: Channel 3 event active

Write:

0bXXX0: No effect  
 0bXXX1: Set event on Channel 0  
 0bXX0X: No effect  
 0bXX1X: Set event on Channel 1  
 0bX0XX: No effect  
 0bX1XX: Set event on Channel 2  
 0b0XXX: No effect  
 0b1XXX: Set event on Channel 3

#### 41.7.4 CTI application trigger clear register (CTI\_APPCLEAR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	APPCLEAR[3:0]														
															w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3:0 **APPCLEAR[3:0]**: Clear channel event.

0b0000: No effect

0bXXX1: Clear event on Channel 0

0bXX1X: Clear event on Channel 1

0bX1XX: Clear event on Channel 2

0b1XXX: Clear event on Channel 3

#### 41.7.5 CTI application pulse register (CTI\_APPPULSER)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	APPULSE[3:0]														
															w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3:0 **APPULSE[3:0]**: Pulse channel event. This register clears itself immediately.

0b0000: No effect

0bXXX1: Generate pulse on Channel 0

0bXX1X: Generate pulse on Channel 1

0bX1XX: Generate pulse on Channel 2

0b1XXX: Generate pulse on Channel 3

#### 41.7.6 CTI trigger In x enable register (CTI\_INENRx)

Address offset: 0x020 + 4 \* x, where x = 0 to 7

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TRIGINEN[3:0]														
															rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3:0 **TRIGINEN[3:0]**: Enables or disables a cross trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).

0b0000: Trigger does not generate events on Channels

0bXXX1: Trigger n generates events on Channel 0

0bXX1X: Trigger n generates events on Channel 1

0bX1XX: Trigger n generates events on Channel 2

0b1XXX: Trigger n generates events on Channel 3

#### 41.7.7 CTI trigger out x enable register (CTI\_OUTENRx)

Address offset: 0x0A0 + 4 \* x, where x = 0 to 7

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
														rw	rw
														rw	rw
TRIGOUTEN[3:0]															

Bits 31:4 Reserved, must be kept at reset value.

Bit 3:0 **TRIGOUTEN[3:0]**: For each channel, defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).

0b0000: Channel events do not generate triggers on Trigger outputs

0bXXX1: Channel 0 events generate triggers on Trigger output n

0bXX1X: Channel 1 events generate triggers on Trigger output n

0bX1XX: Channel 2 events generate triggers on Trigger output n

0b1XXX: Channel 3 events generate triggers on Trigger output n

#### 41.7.8 CTI trigger in status register (CTI\_TRGISTSR)

Address offset: 0x130

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
														r	r
														r	r
														r	r
TRIGINSTATUS[7:0]															

Bits 31:8 Reserved, must be kept at reset value.

Bit 7:0 **TRIGINSTATUS[7:0]**: Trigger input status.

There is one bit of the register for each CTITRIGIN input. When a bit is set to 1 it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

### 41.7.9 CTI trigger out status register (CTI\_TRGOSTSR)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bits 31:8 Reserved, must be kept at reset value.

Bit 7:0 **TRIGOUTSTATUS[7:0]**: Trigger output status.

There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1 it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

### 41.7.10 CTI channel in status register (CTI\_CHINSTSR)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r												

Bits 31:4 Reserved, must be kept at reset value.

Bit 3:0 **CHINSTATUS[3:0]**: Channel input status.

There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

### 41.7.11 CTI channel out status register (CTI\_CHOUTSTS)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CHOUTSTATUS[3:0]														
														r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3:0 **CHOUTSTATUS[3:0]**: Channel output status.

There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

#### 41.7.12 CTI channel gate register (CTI\_GATER)

Address offset: 0x140

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GATEEN[3:0]														
														rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3:0 **GATEEN[3:0]**: Channel output enable. For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.

0b0000: Channels events do not propagate

0bXXX1: Channel 0 events propagate

0bXX1X: Channel 1 events propagate

0bX1XX: Channel 2 events propagate

0b1XXX: Channel 3 events propagate

#### 41.7.13 CTI claim tag set register (CTI\_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
														rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:

0000: No effect  
xxx1: Set bit 0  
xx1x: Set bit 1  
x1xx: Set bit 2  
1xxx: Set bit 3

Read:

0xF: Indicates there are four bits in claim tag

#### 41.7.14 CTI claim tag clear register (CTI\_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
														rw	rw
														rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:

0000: No effect  
xxx1: Clear bit 0  
xx1x: Clear bit 1  
x1xx: Clear bit 2  
1xxx: Clear bit 3

Read: Returns current value of claim tag

#### 41.7.15 CTI lock access register (CTI\_LAR)

Address offset: 0xFB0

Reset value: N/A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS\_W[31:0]**: Enables write access to some CTI registers by processor cores  
(debuggers do not need to unlock the component)

0xC5AC CE55: Write access enabled

Other values: Write access disabled

#### 41.7.16 CTI lock status register (CTI\_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LOCK TYPE	LOCK GRANT	LOCK EXIST												

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Indicates the size of the CTI\_LAR register

0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock. This bit always reads as zero by an external debugger.

0: Write access is permitted

1: Write access is blocked. Only reads are permitted.

Bit 0 **LOCKEXIST**: Indicates whether a lock control mechanism exists. This bit always reads as zero by an external debugger.

0: No lock control mechanism exists.

1: Lock control mechanism is implemented

#### 41.7.17 CTI authentication status register (CTI\_AUTHSTATR)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SNID[1:0]	SID[1:0]	NSNID[1:0]	NSID[1:0]	r	r	r	r							

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug

0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug  
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug  
0x2: Disabled  
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug  
0x2: Disabled  
0x3: Enabled

#### 41.7.18 CTI device configuration register (CTI\_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0004 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMTRIG[7:0]										Res.	Res.	Res.	EXTMUXNUM[4:0]		
r	r	r	r	r	r	r	r					r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NUMCH[3:0]**: Number of ECT channels available  
0x4: 4 channels

Bits 15:8 **NUMTRIG[7:0]**: Number of ECT triggers available  
0x8: 8 trigger inputs and 8 trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]**: Number of trigger input/output multiplexers  
0x0: None

#### 41.7.19 CTI device type identifier register (CTI\_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]										
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification

0x1: Indicates that this component is a cross-triggering component.

Bits 3:0 **MAJORTYPE[3:0]**: Major classification

0x4: Indicates that this component allows a debugger to control other components in a CoreSight™ SoC-400 system.

#### 41.7.20 CTI CoreSight peripheral identity register 4 (CTI\_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	4KCOUNT[3:0]				JEP106CON[3:0]										
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size

0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x4: Arm® JEDEC code

#### 41.7.21 CTI CoreSight peripheral identity register 0 (CTI\_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PARTNUM[7:0]														
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]

0x06: CTI part number

#### 41.7.22 CTI CoreSight peripheral identity register 1 (CTI\_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]

0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]

0x9: CTI part number

#### 41.7.23 CTI CoreSight peripheral identity register 2 (CTI\_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 004B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bit 7:4 **REVISION[3:0]**: Component revision number

0x4: r0p5

Bit 3 **JEDEC**: JEDEC assigned value

0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]

0x3: Arm® JEDEC code

#### 41.7.24 CTI CoreSight peripheral identity register 3 (CTI\_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	r	r	r	r	r	r	r
REVAND[3:0]								CMOD[3:0]							

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version

0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified

0x0: No customer modifications

#### 41.7.25 CTI CoreSight component identity register 0 (CTI\_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	r	r	r	r	r	r	r
PREAMBLE[7:0]															

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]

0x0D: Common ID value

#### 41.7.26 CTI CoreSight peripheral identity register 1 (CTI\_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	r	r	r	r	r	r	r
CLASS[3:0]								PREAMBLE[11:8]							

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class  
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]  
0x0: Common ID value

#### 41.7.27 CTI CoreSight component identity register 2 (CTI\_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREAMBLE[19:12]														
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]  
0x05: Common ID value

#### 41.7.28 CTI CoreSight component identity register 3 (CTI\_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREAMBLE[27:20]														
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]  
0xB1: Common ID value

### 41.7.29 CTI register map and reset values

Table 270. CTI register map and reset values

Offset	Register name	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CTI_CONTROLR		Res.	Res.	0																													
	Reset value																														GLBEN	0		
0x010	CTI_INTACKR		Res.	INTACK[7:0]																														
	Reset value																																	
0x014	CTI_APPSETR		Res.	APPSET[3:0]																														
	Reset value																																	
0x018	CTI_APPCLEAR		Res.	APPCLEAR[3:0]																														
	Reset value																																	
0x01C	CTI_APPPULSER		Res.	APPpulse[3:0]																														
	Reset value																																	
0x020	CTI_INENR0		Res.	TRIGINEN[3:0]																														
	Reset value																																	
0x024	CTI_INENR1		Res.	TRIGINEN[3:0]																														
	Reset value																																	
0x028	CTI_INENR2		Res.	TRIGINEN[3:0]																														
	Reset value																																	
0x02C	CTI_INENR3		Res.	TRIGINEN[3:0]																														
	Reset value																																	
0x030	CTI_INENR4		Res.	TRIGINEN[3:0]																														
	Reset value																																	
0x034	CTI_INENR5		Res.	TRIGINEN[3:0]																														
	Reset value																																	
0x038	CTI_INENR6		Res.	TRIGINEN[3:0]																														
	Reset value																																	
0x03C	CTI_INENR7		Res.	TRIGINEN[3:0]																														
	Reset value																																	
0x0A0	CTI_OUTENR0		Res.	TRIGOUTEN[3:0]																														
	Reset value																																	
0x0S4	CTI_OUTENR1		Res.	TRIGOUTEN[3:0]																														
	Reset value																																	
0x0S8	CTI_OUTENR2		Res.	TRIGOUTEN[3:0]																														
	Reset value																																	

**Table 270. CTI register map and reset values (continued)**

Table 270. CTI register map and reset values (continued)

Offset	Register name	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	CTI_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106CON [3:0]			
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
0xFE0	CTI_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]			
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
0xFE4	CTI_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]	PARTNUM [11:8]		
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1 0 1 1 1 0 0 1 1 0 1		
0xFE8	CTI_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]	JEP106ID [6:4]	
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0 1 0 0 1 0 1 1 0 1		
0xFEC	CTI_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]	CMOD[3:0]	
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0 0 0 0 0 0 0 0 0 0		
0xFF0	CTI_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]		
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0 0 0 0 1 1 0 1 0 1		
0xFF4	CTI_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]	PREAMBLE[11:8]	
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1 0 0 1 0 0 0 0 0 0		
0xFF8	CTI_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]		
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0 0 0 0 0 0 1 0 1 0		
0xFFC	CTI_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]		
	Reset value	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1 0 1 1 0 0 0 0 0 1		

Refer to [Section 41.9: CPU2 ROM tables](#) and to [Section 41.13: CPU1 ROM table](#) for the register boundary addresses.

## 41.8 Microcontroller debug unit (DBGMCU)

The DBGMCU is a component containing a number of registers that control the power and clock behavior in debug mode. It allows the debugger (or the debug software) to:

- maintain the clock and power the CPU1 processor core when in low power modes (Sleep, Stop or Standby), CPU2 operation is not influenced in low power debug mode
- maintain the clock and power the system debug and trace components when in low power modes
- stop the clock to certain peripherals (watchdogs, timers, RTC) when either processor core is stopped in debug mode

The DBGMCU registers are not reset by a system reset, only by a power on reset. They are accessible to the debugger via the CPU1 AHB access port at base address 0xE0042000.

**Note:** *The DBGMCU is not a standard CoreSight™ component, consequently it does not appear in the CPU1 ROM table.*

### 41.8.1 DBGMCU identity code register (DBGMCU\_IDCODE)

Address offset: 0x000

Reset value: 0x200X 6495

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV\_ID[15:0]**: Revision

0x2001 = STM32WB55xx revision Y and STM32WB35xx revision A

0x2003 = STM32WB55xx/35xx revision X

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV\_ID[11:0]**: Device ID

0x495: STM32WB55xx/35xx

### 41.8.2 DBGMCU configuration register (DBGMCU\_CR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TRGOEN	Res.	Res.	Res.	Res.	Res.	Res.						
			<b>rw</b>												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE _IOEN	Res.	Res.	DBG _STANDBY	DBG _STOP	DBG _SLEEP
										<b>rw</b>			<b>rw</b>	<b>rw</b>	<b>rw</b>

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TRGOEN**: External trigger out/put enable. This bit controls the direction of the bi-directional trigger pin, TRIG\_INOUT.

- 0: Input. TRIG\_INOUT is connected to TRGIN.
- 1: Output. TRIG\_INOUT is connected to TRGOUT.

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **TRACE\_IOEN**: Trace port and clock enable. This bit enables the trace port clock, TRACECLK.

- 0: Disabled.
- 1: Enabled.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DBG\_STANDBY**: Allow debug of the CPU1 in STANDBY and SHUTDOWN modes, no influence on CPU2 operation.

- 0: Normal operation. All clocks are disabled and the device powered down automatically in STANDBY and SHUTDOWN modes.
- 1: Automatic clock stop/power down disabled. All active CPU1 clocks and oscillators continue to run during STANDBY and SHUTDOWN modes, and the device supply is maintained, allowing full CPU1 debug capability. On exit from STANDBY and SHUTDOWN modes, a device reset is performed.

*Note: On exit from STANDBY no power reset is performed, a system reset is generated, and the wakeup clock is not HSI16, but MSI when DBG\_STANDBY is set.*

Bit 1 **DBG\_STOP**: Allow debug of the CPU1 in STOP mode, no influence on CPU2 operation.

- 0: Normal operation. All CPU1 clocks are disabled automatically in STOP mode
- 1: Automatic clock stop disabled. All active clocks and oscillators continue to run during STOP mode, allowing full CPU1 debug capability. On exit from STOP mode, the clock settings are set to the STOP mode exit state.

Bit 0 **DBG\_SLEEP**: Allow debug of the CPU1 in SLEEP mode, no influence on CPU2 operation.

- 0: Normal operation. Processor clock is stopped automatically in SLEEP mode
- 1: Automatic clock stop disabled. CPU1 processor clock continue to run, resulting in full CPU1 debug capability

### 41.8.3 DBGMCU CPU1 APB1 peripheral freeze register 1 (DBGMCU\_APB1FZR1)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	Res.	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
rw								rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM2_STOP
			rw	rw	rw										rw

Bit 31 **DBG\_LPTIM1\_STOP**: LPTIM1 stop in CPU1 debug

0: Normal operation. LPTIM1 continues to operate while CPU1 is in debug mode

1: Stop in debug. LPTIM1 is frozen while CPU1 is in debug mode.

Bits 30:24 Reserved, must be kept at reset value.

**DBG\_I2C3\_STOP**: I2C3 SMBUS timeout stop in CPU1 debug

Bit 23 0: Normal operation. I2C3 SMBUS timeout continues to operate while CPU1 is in debug mode

1: Stop in debug. I2C3 SMBUS timeout is frozen while CPU1 is in debug mode.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **DBG\_I2C1\_STOP**: I2C1 SMBUS timeout stop in CPU1 debug

0: Normal operation. I2C1 SMBUS timeout continues to operate while CPU1 is in debug mode

1: Stop in debug. I2C1 SMBUS timeout is frozen while CPU1 is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG\_IWDG\_STOP**: IWDG stop in CPU1 debug

0: Normal operation. IWDG continues to operate while CPU1 is in debug mode

1: Stop in debug. IWDG is frozen while CPU1 is in debug mode.

Bit 11 **DBG\_WWDG\_STOP**: WWDG stop in CPU1 debug

0: Normal operation. WWDG continues to operate while CPU1 is in debug mode

1: Stop in debug. WWDG is frozen while CPU1 is in debug mode.

Bit 10 **DBG\_RTC\_STOP**: RTC stop in CPU1 debug

0: Normal operation. RTC continues to operate while CPU1 is in debug mode

1: Stop in debug. RTC is frozen while CPU1 is in debug mode.

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **DBG\_TIM2\_STOP**: TIM2 stop in CPU1 debug

0: Normal operation. TIM2 continues to operate while CPU1 is in debug mode

1: Stop in debug. TIM2 is frozen while CPU1 is in debug mode.

#### 41.8.4 **DBGMCU CPU2 APB1 peripheral freeze register 1 (DBGMCU\_C2APB1FZR1)**

CPU2 peripheral freeze is only available when CPU2 debug is enabled or when CPU2 CTI is configured by the debugger for halt via EDBGHRQ.

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	Res.	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
rw								rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM2_STOP
			rw		rw										rw

**DBG\_LPTIM1\_STOP:** LPTIM1 stop in CPU2 debug

- Bit 31 0: Normal operation. LPTIM1 continues to operate while CPU2 is in debug mode  
1: Stop in debug. LPTIM1 is frozen while CPU2 is in debug mode.

Bits 30:24 Reserved, must be kept at reset value.

**DBG\_I2C3\_STOP:** I2C3 SMBUS timeout stop in CPU2 debug

- Bit 23 0: Normal operation. I2C3 SMBUS timeout continues to operate while CPU2 is in debug mode  
1: Stop in debug. I2C3 SMBUS timeout is frozen while CPU2 is in debug mode.

Bit 22 Reserved, must be kept at reset value.

**DBG\_I2C1\_STOP:** I2C1 SMBUS timeout stop in CPU2 debug

- Bit 21 0: Normal operation. I2C1 SMBUS timeout continues to operate while CPU2 is in debug mode  
1: Stop in debug. I2C1 SMBUS timeout is frozen while CPU2 is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG\_IWDG\_STOP:** IWDG stop in CPU2 debug

- 0: Normal operation. IWDG continues to operate while CPU2 is in debug mode  
1: Stop in debug. IWDG is frozen while CPU2 is in debug mode.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **DBG\_RTC\_STOP:** RTC stop in CPU2 debug

- 0: Normal operation. RTC continues to operate while CPU2 is in debug mode  
1: Stop in debug. RTC is frozen while CPU2 is in debug mode.

Bits 9:1 Reserved, must be kept at reset value.

Bit 0 **DBG\_TIM2\_STOP:** TIM2 stop in CPU2 debug

- 0: Normal operation. TIM2 continues to operate while CPU2 is in debug mode  
1: Stop in debug. TIM2 is frozen while CPU2 is in debug mode.

#### 41.8.5 DBGMCU CPU1 APB1 peripheral freeze register 2 (DBGMCU\_APB1FZR2)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DBG_LPTIM2_STOP	Res.	Res.	Res.	Res.	Res.									
										rw					

Bits 31:9 Reserved, must be kept at reset value.

Bit 5 **DBG\_LPTIM2\_STOP**: LPTIM2 stop in CPU1 debug

0: Normal operation. LPTIM2 continues to operate while CPU1 is in debug mode

1: Stop in debug. LPTIM2 is frozen while CPU1 is in debug mode.

Bits 7:0 Reserved, must be kept at reset value.

#### 41.8.6 DBGMCU CPU2 APB1 peripheral freeze register 2 (DBGMCU\_C2APB1FZR2)

CPU2 peripheral freeze is only available when CPU2 debug is enabled or when CPU2 CTI is configured by the debugger for halt via EDBGHQ.

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DBG_LPTIM2_STOP	Res.	Res.	Res.	Res.	Res.									
										rw					

Bits 31:9 Reserved, must be kept at reset value.

Bit 5 **DBG\_LPTIM2\_STOP**: LPTIM2 stop in CPU2 debug

0: Normal operation. LPTIM2 continues to operate while CPU2 is in debug mode

1: Stop in debug. LPTIM2 is frozen while CPU2 is in debug mode.

Bits 7:0 Reserved, must be kept at reset value.

#### 41.8.7 DBGMCU CPU1 APB2 peripheral freeze register (DBGMCU\_APB2FZR)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	Res.
															rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Res.	Res.	Res.	Res.	DBG_TIM17_STOP	Res.	Res.	Res.										
				rw													

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG\_TIM17\_STOP**: TIM17 stop in CPU1 debug

0: Normal operation. TIM17 continues to operate while CPU1 is in debug mode

1: Stop in debug. TIM17 is frozen while CPU1 is in debug mode.

Bit 17 **DBG\_TIM16\_STOP**: TIM16 stop in CPU1 debug

0: Normal operation. TIM16 continues to operate while CPU1 is in debug mode

1: Stop in debug. TIM16 is frozen while CPU1 is in debug mode.

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **DBG\_TIM1\_STOP**: TIM1 stop in CPU1 debug

0: Normal operation. TIM1 continues to operate while CPU1 is in debug mode

1: Stop in debug. TIM1 is frozen while CPU1 is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

#### 41.8.8 DBGMCU CPU2 APB2 peripheral freeze register (DBGMCU\_C2APB2FZR)

CPU2 peripheral freeze is only available when CPU2 debug is enabled or when CPU2 CTI is configured by the debugger for halt via EDBGHQ.

Address offset: 0x050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
															rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Res.	Res.	Res.	Res.	DBG_TIM17_STOP	Res.												
				rw													

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG\_TIM17\_STOP**: TIM17 stop in CPU2 debug

0: Normal operation. TIM17 continues to operate while CPU2 is in debug mode

1: Stop in debug. TIM17 is frozen while CPU2 is in debug mode.

Bit 17 **DBG\_TIM16\_STOP**: TIM16 stop in CPU2 debug

- 0: Normal operation. TIM16 continues to operate while CPU2 is in debug mode
- 1: Stop in debug. TIM16 is frozen while CPU2 is in debug mode.

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **DBG\_TIM1\_STOP**: TIM1 stop in CPU2 debug

- 0: Normal operation. TIM1 continues to operate while CPU2 is in debug mode
- 1: Stop in debug. TIM1 is frozen while CPU2 is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

#### 41.8.9 DBGMCU register map and reset values

**Table 271. DBGMCU register map and reset values**

Table 271. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x050	DBGMCU_C2APB2FZR	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														
	Reset value																																

Refer to [Section 41.8: Microcontroller debug unit \(DBGMCU\)](#) for the register boundary addresses.

## 41.9 CPU2 ROM tables

The ROM tables are CoreSight™ components that contain the base addresses of all the Coresight™ debug components accessible via the AHB-D. These tables allow a debugger to discover the topology of the CoreSight™ system automatically.

There are two ROM tables in the CPU2 sub-system:

1. ROM1: CPU2 processor ROM table, pointed to by the BASE register in the CPU2 AHB-AP. It contains the base address pointers for the CTI, as well as for the CPU2 ROM table.
2. ROM2: CPU2 ROM table, containing pointers to the CPU2 System control space registers, which allow the debugger to identify the CPU core, as well as to the remaining CoreSight™ components in the CPU2 subsystem (PBU, DWT).

The CPU2 processor ROM table occupies a 4-Kbyte, 32-bit wide chunk of AHB address space, from 0xF0000000 to 0xF0000FFC.

**Table 272. CPU2 processor ROM table**

Address in ROM table	Component name	Component base address	Component address offset	Size	Entry
0xF0000000	CPU2 ROM table	0xE00FF000	0xF00FF000	4 KB	0xF00FF003
0xF0000004	CTI	0xF0001000	0x00001000	4 KB	0x00001003
0xF0000008	Not used	-	-	-	0x00002002
0xF000000C	Not used	-	-	-	0x10000002
0xF0000010	Top of table	-	-	-	0x00000000
0xF000000C to 0xF0000FC8	Reserved	-	-	-	0x00000000
0xF0000FCC to 0xF0000FFC	ROM table registers	-	-	-	See <a href="#">Table 274</a>

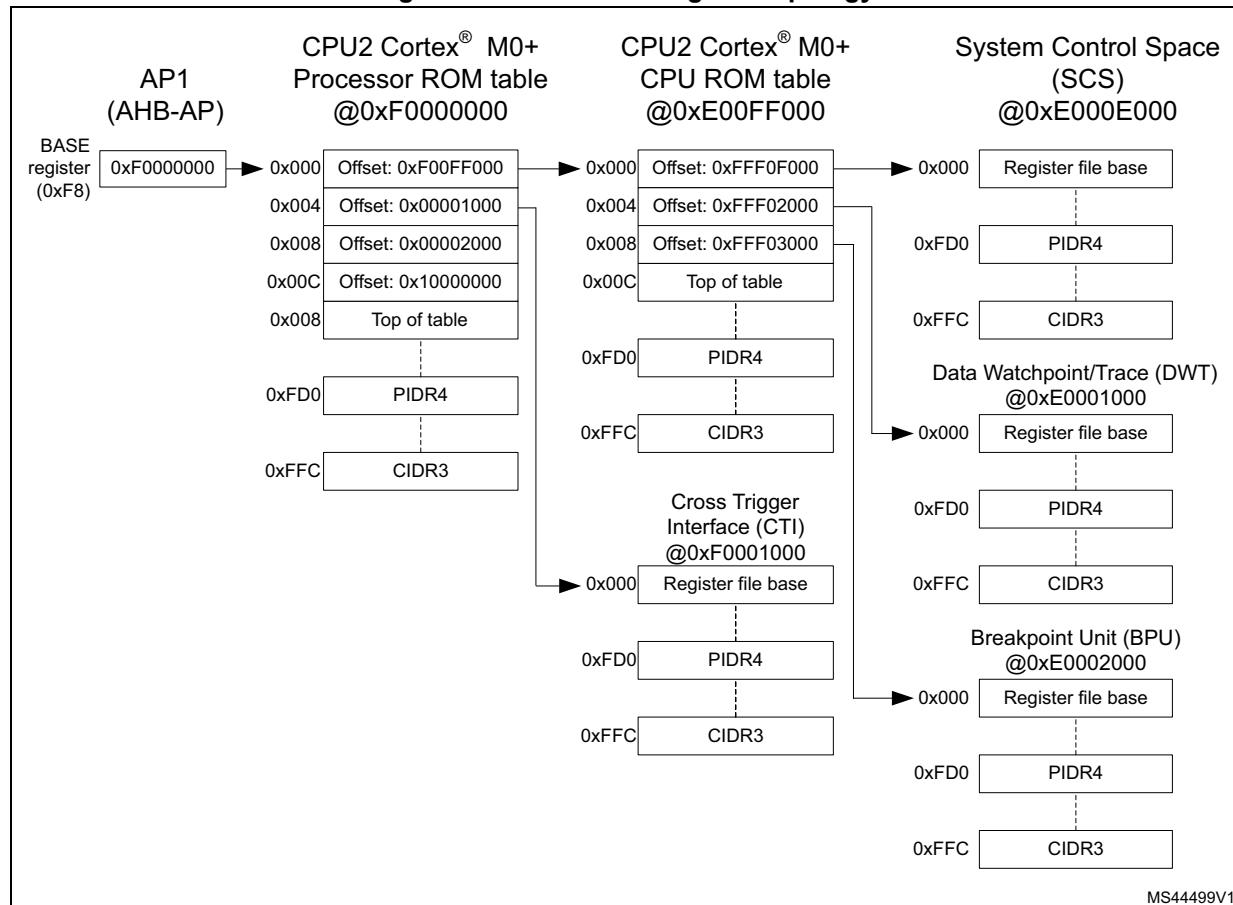
The CPU2 ROM table occupies a 4 KB, 32-bit wide chunk of APB-D address space, from 0xE00FF000 to 0xE00FFFFC.

**Table 273. CPU2 ROM table**

Address in ROM table	Component name	Component base address	Component address offset	Size	Entry
0xE00FF000	SCS	0xE000E000	0xFFFF0F000	4 KB	0xFFFF0F003
0xE00FF004	DWT	0xE0001000	0xFFFF02000	4 KB	0xFFFF02003
0xE00FF008	BPU	0xE0002000	0xFFFF03000	4 KB	0xFFFF03003
0xE00FF00C	Top of table	-	-	-	0x00000000
0xE00FF010 to 0xE00FFFC8	Reserved	-	-	-	0x00000000
0xE00FFFCC to 0xE00FFFFC	ROM table registers	-	-	-	See <a href="#">Table 275</a>

The topology for the CoreSight™ components in the CPU2 subsystem is shown in [Figure 419](#).

**Figure 419. CPU2 CoreSight™ topology**



### 41.9.1 CPU2 ROM1 memory type register (C2ROM1\_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SYSMEM														
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory

0x1: System memory is present on this bus

### 41.9.2 CPU2 ROM1 CoreSight peripheral identity register 4 (C2ROM1\_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	4KCOUNT[3:0]				JEP106CON[3:0]										
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size

0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x4: Arm® JEDEC continuation code

### 41.9.3 CPU2 ROM1 CoreSight peripheral identity register 0 (C2ROM1\_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]

0xC0: Cortex®-M0+ processor ROM table

#### 41.9.4 CPU2 ROM1 CoreSight peripheral identity register 1 (C2ROM1\_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PARTNUM[11:8]							
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]

0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]

0x4: Cortex®-M0+processor ROM table

#### 41.9.5 CPU2 ROM1 CoreSight peripheral identity register 2 (C2ROM1\_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								REVISION[3:0]							
								r	r	r	r	r	r	r	JEDEC
															JEP106ID[6:4]

Bit 31:8 Reserved, must be kept at reset value.

Bit s 7:4 **REVISION[3:0]**: Component revision number  
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value  
0x1: Designer ID specified by JEDEC

bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits  
0x3: Arm® JEDEC code

## 41.9.6 CPU2 ROM1 CoreSight peripheral identity register 3 (C2ROM1\_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVAND[3:0]				CMOD[3:0]										
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified  
0x0: No customer modifications

#### 41.9.7 CPU2 ROM1 CoreSight component identity register 0 (C2ROM1\_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]  
0x0D: Common ID value

### 41.9.8 CPU2 ROM1 CoreSight peripheral identity register 1 (C2ROM1\_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CLASS[3:0]				PREAMBLE[11:8]										
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class  
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]  
0x0: Common ID value

### 41.9.9 CPU2 ROM1 CoreSight component identity register 2 (C2ROM1\_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREAMBLE[19:12]														
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]  
0x05: Common ID value

### 41.9.10 CPU2 ROM1 CoreSight component identity register 3 (C2ROM1\_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	r	r	r	r	r	r	r	r
PREAMBLE[27:20]															

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]

0xB1: Common ID value

#### 41.9.11 CPU2 processor ROM table registers and reset values

**Table 274. CPU2 processor ROM table register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0xFCC	C2ROM1_MEMTYPER	Res.																																			
	Reset value																																	1	0		
0xFD0	C2ROM1_PIDR4	Res.																																			
	Reset value																																0	1	0	0	
0xFE0	C2ROM1_PIDR0	Res.																																			
	Reset value																																	0	1	0	0
0xFE4	C2ROM1_PIDR1	Res.																																			
	Reset value																																1	1	0	0	0
0xFE8	C2ROM1_PIDR2	Res.																																			
	Reset value																																1	0	1	0	0
0xFEC	C2ROM1_PIDR3	Res.																																			
	Reset value																																0	0	0	0	0
0xFF0	C2ROM1_CIDR0	Res.																																			
	Reset value																																	1	1	0	0
0xFF4	C2ROM1_CIDR1	Res.																																			
	Reset value																																	0	0	0	0
0xFF8	C2ROM1_CIDR2	Res.																																			
	Reset value																																	0	0	0	0
0xFFC	C2ROM1_CIDR3	Res.																																			
	Reset value																																	1	0	1	1

Refer to [Section 41.9: CPU2 ROM tables](#) for the register boundary addresses.

#### 41.9.12 CPU2 ROM2 memory type register (C2ROM2\_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SYSMEM														
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory

0x1: System memory is present on this bus

#### 41.9.13 CPU2 ROM2 CoreSight peripheral identity register 4 (C2ROM2\_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		4KCOUNT[3:0]			JEP106CON[3:0]										
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size

0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x4: Arm® JEDEC continuation code

#### 41.9.14 CPU2 ROM2 CoreSight peripheral identity register 0 (C2ROM2\_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]

0xC0: CPU2 ROM table

#### 41.9.15 CPU2 ROM2 CoreSight peripheral identity register 1 (C2ROM2\_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]

0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]

0x4: CPU2 ROM table

#### 41.9.16 CPU2 ROM2 CoreSight peripheral identity register 2 (C2ROM2\_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bit 31:8 Reserved, must be kept at reset value.

Bit 7:4 **REVISION[3:0]**: Component revision number  
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value  
0x1: Designer ID specified by JEDEC  
Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]  
0x3: Arm® JEDEC code

#### 41.9.17 CPU2 ROM2 CoreSight peripheral identity register 3 (C2ROM2\_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVAND[3:0]				CMOD[3:0]										
									r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified  
0x0: No customer modifications

#### 41.9.18 CPU2 ROM2 CoreSight component identity register 0 (C2ROM2\_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREAMBLE[7:0]														
									r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]  
0x0D: Common ID value

### 41.9.19 CPU2 ROM2 CoreSight peripheral identity register 1 (C2ROM2\_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
								r	r	r	r	r	r	r	r
CLASS[3:0]										PREAMBLE[11:8]					

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class  
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]  
0x0: Common ID value

### 41.9.20 CPU2 ROM2 CoreSight component identity register 2 (C2ROM2\_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
								r	r	r	r	r	r	r	r
PREAMBLE[19:12]															

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]  
0x05: Common ID value

### 41.9.21 CPU2 ROM2 CoreSight component identity register 3 (C2ROM2\_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[27:20]							
								3r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]

0xB1: Common ID value

### 41.9.22 CPU2 ROM table register map and reset values

Table 275. CPU2 ROM table register map and reset values

Offset	Register name	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0xFCC	C2ROM2_MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	SYSMEM	0					
	Reset value																																					
0xFD0	C2ROM2_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	0	1	0	0
	Reset value																																					
0xFE0	C2ROM2_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	0	1	0	0
	Reset value																																					
0xFE4	C2ROM2_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	1	0	0	0	0	0	0
	Reset value																																					
0xFE8	C2ROM2_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	0	1	0	0
	Reset value																																					
0xFEC	C2ROM2_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	1	0	0	0	0	0	0
	Reset value																																					
0xFF0	C2ROM2_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	0	0	0	0
	Reset value																																					
0xFF4	C2ROM2_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	1	1	0	1
	Reset value																																					
0xFF8	C2ROM2_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	0	0	1	0
	Reset value																																					
0xFFC	C2ROM2_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	0	1	1	0	0	0	1
	Reset value																																					

Refer to [Section 41.9: CPU2 ROM tables](#) for the register boundary addresses.

## 41.10 CPU2 data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as:

- Watchpoint
- ETM trigger, only available on STM32WB55xx
- PC sampling trigger
- Data address sampling trigger
- Data comparator (for comparator 1 only)
- Clock cycle counter comparator (for comparator 0 only)

It also contains counters for:

- Clock cycles
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction
- Interrupt overhead

A DWT comparator compares the value held in its DWT\_COMPR register with one of the following items:

- a data address
- an instruction address
- a data value
- the cycle count value (comparator 0 only)

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- One or more DWT Data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value.
- A watchpoint debug event, on either the PC value or the accessed data address.
- A CMPMATCH[N] event, that signals the match outside the DWT unit.

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v7-M Architecture Reference Manual [\[5\]](#).

### 41.10.1 DWT control register (DWT\_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
				NOTRCPKT	NOEXTTRIG	NOCYC CNT	NOPRFCNT	Res.	CYCEVT ENA	FOLDEV TENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTE NA	EXCTR CENA
r	r	r	r	r	r	r	r		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAM PLENA	SYNCTAP[1:0]	CYCTAP				POSTINIT[3:0]				POSTRESET[3:0]		CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP**: Number of comparators implemented (read only)

0x4: Four comparators

Bit 27 **NOTRCPKT**: Trace sampling and exception tracing support (read only)

0x0: Supported

Bit 26 **NOEXTTRIG**: External match signal, CMPMATCH support (read only)

0x0: Supported

Bit 25 **NOCYC CNT**: Cycle counter support (read only)

0x0: Supported

Bit 24 **NOPRFCNT**: Profiling counter support (read only)

0x0: Supported

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CYCEVTENA**: Enable for POSTCNT underflow event counter packet generation

0x0: Disabled

0x1: Enabled

Bit 21 **FOLDEV TENA**: Enable for folded instruction counter overflow event generation

0x0: Disabled

0x1: Enabled

Bit 20 **LSUEVTENA**: Enable for LSU counter overflow event generation

0x0: Disabled

0x1: Enabled

Bit 19 **SLEEPEVTENA**: Enable for sleep counter overflow event generation

0x0: Disabled

0x1: Enabled

Bit 18 **EXCEVTENA**: Enable for exception overhead counter overflow event generation

0x0: Disabled

0x1: Enabled

Bit 17 **CPIEVTE NA**: Enable for CPI counter overflow event generation

0x0: Disabled

0x1: Enabled

Bit 16 **EXCTR CENA**: Enable for exception trace generation

0x0: Disabled

0x1: Enabled

Bits 15:13 Reserved, must be kept at reset value.

- Bit 12 **PCSAMPLENA**: Enables use of POSTCNT counter as a timer for Periodic PC sample packet generation.  
 0x0: Disabled  
 0x1: Enabled
- Bits 11:10 **SYNCTAP[1:0]**: Selects the position of the synchronization packet counter tap on the CYCCNT counter. This determines the synchronization packet rate.  
 0x0: Disabled. No synchronization packets  
 0x1: Tap at CYCCNT[24]  
 0x2: Tap at CYCCNT[26]  
 0x3: Tap at CYCCNT[28]
- Bit 9 **CYCTAP**: Selects the position of the POSTCNT tap on the CYCCNT counter.  
 0x0: Tap at CYCCNT[6]  
 0x1: Tap at CYCCNT[10]
- Bits 8:5 **POSTINIT[3:0]**: Initial value of the POSTCNT counter. Writes to this field are ignored if POSTCNT counter is enabled (ie. CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).
- Bits 4:1 **POSTPRESET[3:0]**: Reload value of the POSTCNT counter.
- Bit 0 **CYCCNTENA**: Enables CYCCNT counter.  
 0x0: Disabled  
 0x1: Enabled

#### 41.10.2 DWT cycle count register (DWT\_CYCCNTR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCCNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CYCCNT[31:0]**: Processor clock cycle counter

#### 41.10.3 DWT CPI count register (DWT\_CPICNTR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	rw						
CPICNT[7:0]															

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]**: CPI counter. Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT\_LSUCNTR, and counts any instruction fetch stalls.

#### 41.10.4 DWT exception count register (DWT\_EXCCNTR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	rw						
EXCCNT[7:0]															

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: Exception overhead cycle counter. Counts the number of cycles spent in exception processing.

#### 41.10.5 DWT sleep count register (DWT\_SLP CNTR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	rw						
SLEEP CNT[7:0]															

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEP CNT[7:0]**: Sleep cycle counter. Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

#### 41.10.6 DWT LSU count register (DWT\_LSUCNTR)

Address offset: 0x014

Reset value: 0x0000 0000

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]**: Load store counter. Counts additional cycles required to execute load and store instructions.

#### 41.10.7 DWT fold count register (DWT\_FOLDCNTR)

Address offset: 0x018

Reset value: 0x0000 0000

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]**: Folded instruction counter. Increments on each instruction that takes 0 cycles.

#### 41.10.8 DWT program counter sample register (DWT\_PCSR)

Address offset: 0x01C

Reset value: 0x0000 0000

Bits 31:0 **EIASAMPLE[31:0]**: Executed Instruction Address sample value. Samples the current value of the program counter.

#### 41.10.9 DWT comparator register x (DWT\_COMPxR)

Address offset:  $0x020 + x * 0x10$  (for  $x = 0$  to  $3$ )

Reset value: 0x0000 0000

Bits 31:0 **COMP[31:0]**: Reference value for comparison.

#### 41.10.10 DWT mask register x (DWT\_MASKxR)

Address offset:  $0x024 + x * 0x10$  (for  $x = 0$  to  $3$ )

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
MASK[4:0]															
												rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **MASK[4:0]**: Comparator mask size. Provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

#### 41.10.11 DWT function register x (DWT\_FUNCTxR)

Address offset:  $0x028 + x * 0x10$  (for  $x = 0$  to  $3$ )

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]					
							r					rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
DATAVADDR0[3:0]				DATAVSIZE[1:0]		LINK1 ENA	DATAV MATCH	CYC MATCH	Res.	EMIT RANGE	Res.	FUNCTION[3:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw		

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match (read only). Indicates if a comparator match has occurred since the register was last read.

0: No match

1: Match occurred

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **DATAVADDR1[3:0]**: When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a second comparator to use for linked address comparison.

Bits 15:12 **DATAVADDR0[3:0]**: When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a comparator to use for linked address comparison.

Bits 11:10 **DATAVSIZE[1:0]**: For data value matching, specifies the size of the required data comparison.

0x0: Byte

0x1: Half word

0x2: Word

0x3: reserved

Bit 9 **LINK1ENA**: Indicates whether use of a second linked comparator is supported (read only).

0x1: Supported

Bit 8 **DATAVMATCH**: Enables cycle comparison.

0x0: Perform address comparison

0x1: Perform data value comparison

Bit 7 **CYCMATCH**: Enables cycle count comparison on comparator 0. This field is reserved for other comparators.

0x0: No cycle count comparison

0x1: Compare DWT\_COMP0R with the cycle counter, DWT\_CYCCNTR

Bit 6 Reserved, must be kept at reset value.

Bit 5 **EMITRANGE**: Enables generation of data trace address offset packets (containing data address bits 0 to 15)

0x0: Disabled

0x1: Enabled

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **FUNCTION[3:0]**: Selects action to take on comparator match. The meaning of this bit field depends on the setting of the DATAVMATCH and CYCMATCH fields. See [\[5\]](#).

#### 41.10.12 DWT CoreSight peripheral identity register 4 (DWT\_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size

0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x4: Arm® JEDEC code

#### 41.10.13 DWT CoreSight peripheral identity register 0 (DWT\_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]

0x02: DWT part number

#### 41.10.14 DWT CoreSight peripheral identity register 1 (DWT\_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]  
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]  
0x9: DWT part number

#### 41.10.15 DWT CoreSight peripheral identity register 2 (DWT\_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]									
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bit 7:4 **REVISION[3:0]**: Component revision number  
0x3: r0p4

Bit 3 **JEDEC**: JEDEC assigned value  
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]  
0x3: Arm® JEDEC code

#### 41.10.16 DWT CoreSight peripheral identity register 3 (DWT\_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVAND[3:0]				CMOD[3:0]										
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified  
0x0: No customer modifications

#### 41.10.17 DWT CoreSight component identity register 0 (DWT\_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	r	r	r	r	r	r	r
PREAMBLE[7:0]															

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]

0x0D: Common ID value

#### 41.10.18 DWT CoreSight peripheral identity register 1 (DWT\_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	r	r	r	r	r	r	r
CLASS[3:0]															
PREAMBLE[11:8]															

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class

0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]

0x0: Common ID value

#### 41.10.19 DWT CoreSight component identity register 2 (DWT\_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]

0x05: Common ID value

#### 41.10.20 DWT CoreSight component identity register 3 (DWT\_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]

0xB1: Common ID value

### 41.10.21 CPU2 DWT registers

The CPU2 DWT registers are located at address range 0xE0001000 to 0xE0001FFC, on the AHBD.

**Table 276. CPU2 DWT register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DWT_CTRLR																																
		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	0	
0x004	DWT_CYCCNTR																																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x008	DWT_CPICNTR																																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x00C	DWT_EXCCNTR																																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x010	DWT_SLPICNTR																																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x014	DWT_LSUCNTR																																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x018	DWT_FOLDCNTR																																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x01C	DWT_PCSR																																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x020	DWT_COMP0R																																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x024	DWT_MASK0R																																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x028	DWT_FUNCT0R																																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x030	DWT_COMP1R																																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x034	DWT_MASK1R																																
		0	0	0	0	0	0	0	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 276. CPU2 DWT register map and reset values (continued)**

Table 276. CPU2 DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF8	DWT_CIDR2	Res.	PREAMBLE[19:12]																														
	Reset value	0	0	0	0	0	0	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	0
0xFFC	DWT_CIDR3	Res.	PREAMBLE[27:20]																														
	Reset value	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	

Refer to [Section 41.9: CPU2 ROM tables](#) for the register boundary addresses.

## 41.11 CPU2 breakpoint unit (PBU)

The BPU allows the user to set hardware breakpoints. It contains eight comparators that monitor the instruction fetch address and return a breakpoint instruction when a match is detected. The CPU2 PBU does not support Flash memory patch functionality.

### 41.11.1 BPU control register (BPU\_CTRLR)

Address offset: 0x000

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE[6:4]			NUM_LIT[3:0]				NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	r	r	r	r	r	r	r	r	r	r	r			rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **NUM\_CODE[6:4]**: Number of instruction address comparators supported - least significant bits (read only).

0x0: 8 instruction comparators supported.

Bits 11:8 **NUM\_LIT[3:0]**: Number of literal address comparators supported (read only).

0x0: No literal comparators supported.

Bits 7:4 **NUM\_CODE[3:0]**: Number of instruction address comparators supported - least significant bits (read only).

0x8: 8 instruction comparators supported

Bit 1 **KEY**: Write protect key. A write to BPU\_CTRLR register is ignored if this bit is not set to 1.

Bits 0 **ENABLE**: BPU enable

0x0: Disable

0x1: Enable

### 41.11.2 BPU remap register (BPU\_REMAPR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	RMPSP	Res.												
		r													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **RMPSP**: Indicates whether Flash memory patch remap is supported (read only).  
0x0: Remapping not supported.

Bits 28:0 Reserved, must be kept at reset value.

### 41.11.3 BPU comparator registers (BPU\_COMPxR)

Address offset: 0x008 + x \* 0x4 (for x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REPLACE[1:0]	Res.	COMP[26:14]													
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
COMP[13:0]														Res.	ENABLE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **REPLACE[1:0]**: Defines the behavior when a match occurs between the COMP field and the instruction fetch address.

0x0: Reserved

0x1: Breakpoint on lower half-word, upper half-word is unaffected.

0x2: Breakpoint on upper half-word, lower half-word is unaffected.

0x3: Breakpoint on both upper and lower half-words.

Bit 29 Reserved, must be kept at reset value.

Bits 28:2 **COMP[26:0]**: Value to compare with address bits 28:2 of accesses to instruction code memory (0x00000000 to 0x1FFFFFFF). If a match occurs, the action to be taken is defined by the REPLACE field.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Comparator enable. The comparator is only enabled if both this bit and the BPU\_ENABLE bit in the BPU\_CTRLR register are set.

0: Disabled

1: Enabled

### 41.11.4 BPU CoreSight peripheral identity register 4 (BPU\_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	4KCOUNT[3:0]				JEP106CON[3:0]										
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

- Bits 7:4 **4KCOUNT[3:0]**: register file size  
 0x0: Register file occupies a single 4 KB region
- Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code  
 0x4: Arm® JEDEC code

#### 41.11.5 BPU CoreSight peripheral identity register 0 (BPU\_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PARTNUM[7:0]														
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

- Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]  
 0x0C: BPU part number

#### 41.11.6 BPU CoreSight peripheral identity register 1 (BPU\_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	JEP106ID[3:0]				PARTNUM[11:8]										
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

- Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]  
 0xB: Arm® JEDEC code
- Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]  
 0x0: BPU part number

#### 41.11.7 BPU CoreSight peripheral identity register 2 (BPU\_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]									
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bit 7:4 **REVISION[3:0]**: Component revision number

0x2: r0p3

Bit 3 **JEDEC**: JEDEC assigned value

0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]

0x3: Arm® JEDEC code

#### 41.11.8 BPU CoreSight peripheral identity register 3 (BPU\_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVAND[3:0]				CMOD[3:0]										
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version

0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified

0x0: No customer modifications

#### 41.11.9 BPU CoreSight component identity register 0 (BPU\_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]

0x0D: Common ID value

#### 41.11.10 BPU CoreSight peripheral identity register 1 (BPU\_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class

0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]

0x0: Common ID value

#### 41.11.11 BPU CoreSight component identity register 2 (BPU\_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]  
0x05: Common ID value

#### 41.11.12 BPU CoreSight component identity register 3 (BPU\_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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
PREAMBLE[27:20]															
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]  
0xB1: Common ID value

### 41.11.13 CPU2 BPU register map and reset values

The CPU2 BPU registers are located at address range 0xE0002000 to 0xE0002FFC.

**Table 277. CPU2 BPU register map and reset values**

Refer to [Section 41.9: CPU2 ROM tables](#) for the register boundary addresses.

## 41.12 CPU2 cross trigger interface (CTI)

See [Section 41.6](#).

## 41.13 CPU1 ROM table

The ROM table is a CoreSight™ component that contains the base addresses of all the Coresight™ debug components accessible via the AHB-AP. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There is one ROM table in the CPU1 sub-system. This table is pointed to by the BASE register in the CPU1 AHB-AP. It contains the base address pointer for the System control space registers, which allows the debugger to identify the CPU core, as well as for the FPB, DWT, ITM, ETM and CTI.

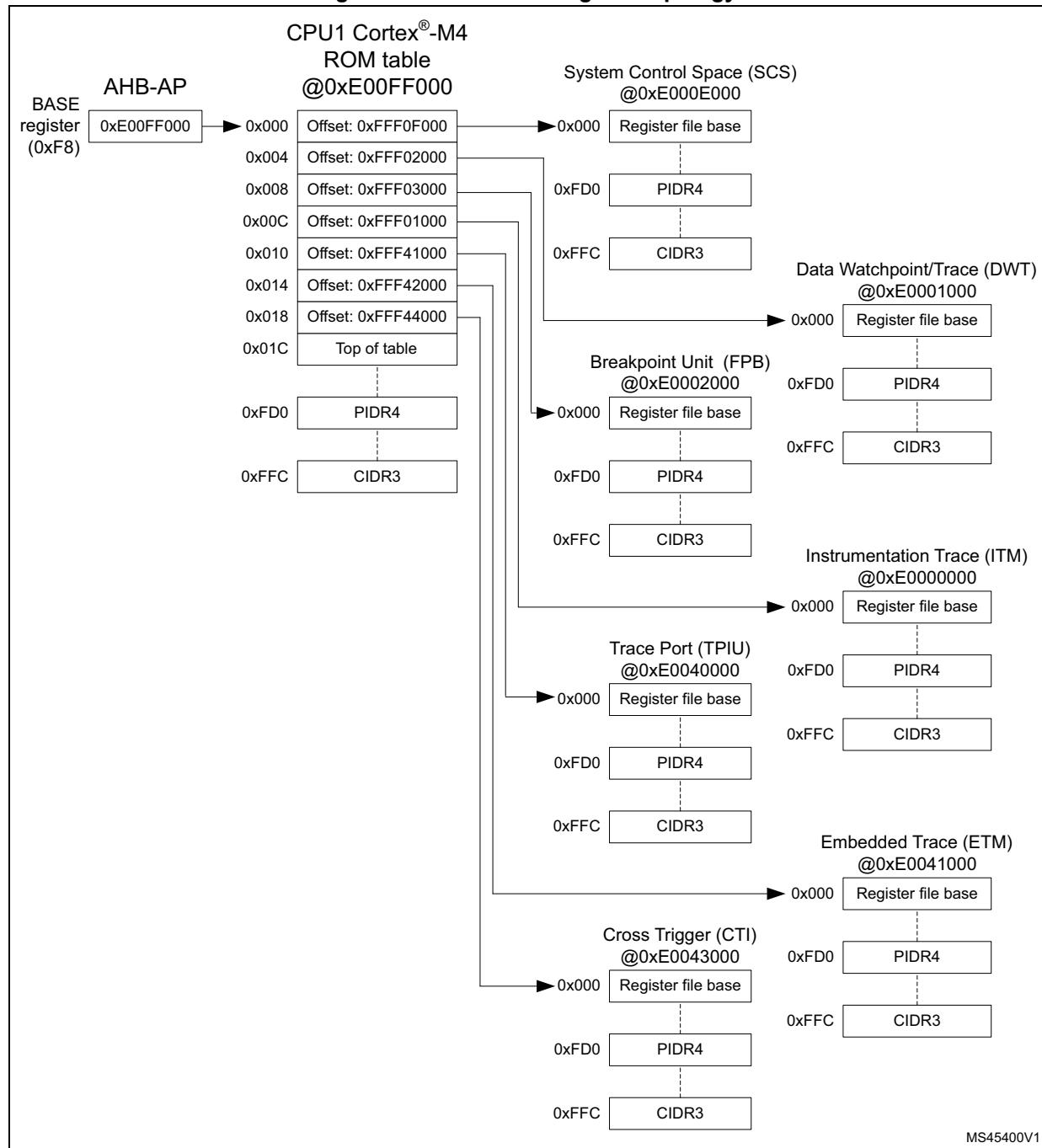
The CPU1 ROM table (see [Table 278](#)) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00FF000 to 0xE00FFFFC.

**Table 278. CPU1 ROM table**

Address in ROM table	Component name	Component base address	Component address offset	Size	Entry
0xE00FF000	SCS	0xE000E000	0xFFFF0F000	4 KB	0xFFFF0F003
0xE00FF004	DWT	0xE0001000	0xFFFF02000	4 KB	0xFFFF02003
0xE00FF008	FPB	0xE0002000	0xFFFF03000	4 KB	0xFFFF03003
0xE00FF00C	ITM	0xE0000000	0xFFFF01000	4 KB	0xFFFF01003
0xE00FF010	TPIU	0xE0040000	0xFFFF41000	4 KB	0xFFFF41003
0xE00FF014	ETM	0xE0041000	0xFFFF42000	4 KB	0xFFFF42003
0xE00FF018	CTI	0xE0043000	0xFFFF44000	4 KB	0xFFFF44003
0xE00FF01C	Top of table	-	-	-	0x00000000
0xE00FF020 to 0xE00FFFC8	Reserved	-	-	-	0x00000000
0xE00FFFCC to 0xE00FFFFC	ROM table registers	-	-	-	See <a href="#">Table 274</a>

The topology for the CoreSight™ components in the CPU1 subsystem is shown in [Figure 420](#).

Figure 420. CPU1 CoreSight™ topology



#### 41.13.1 CPU1 ROM memory type register (C1ROM\_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SYSMEM														
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory

0x1: System memory is present on this bus

### 41.13.2 CPU1 ROM CoreSight peripheral identity register 4 (C1ROM\_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	JEP106CON[3:0]								
									r	r	r	r	r	r	

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size

0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x0: STMicroelectronics JEDEC continuation code

### 41.13.3 CPU1 ROM CoreSight peripheral identity register 0 (C1ROM\_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0095

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	PARTNUM[7:0]								
									r	r	r	r	r	r	

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]  
0x95: STM32WB55xx/35xx

#### 41.13.4 CPU1 ROM CoreSight peripheral identity register 1 (C1ROM\_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]  
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]  
0x4: STM32WB55xx/35xx

#### 41.13.5 CPU1 ROM CoreSight peripheral identity register 2 (C1ROM\_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number  
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value

0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]  
0x2: STMicroelectronics JEDEC code

#### 41.13.6 CPU1 ROM CoreSight peripheral identity register 3 (C1ROM\_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version

0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified

0x0: No customer modifications

#### 41.13.7 CPU1 ROM CoreSight component identity register 0 (C1ROM\_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]

0x0D: Common ID value

#### 41.13.8 CPU1 ROM CoreSight peripheral identity register 1 (C1ROM\_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		CLASS[3:0]		PREAMBLE[11:8]											
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class  
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]  
0x0: Common ID value

#### 41.13.9 CPU1 ROM CoreSight component identity register 2 (C1ROM\_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		PREAMBLE[19:12]													
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]  
0x05: Common ID value

#### 41.13.10 CPU1 ROM CoreSight component identity register 3 (C1ROM\_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		PREAMBLE[27:20]													
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]  
0xB1: Common ID value

### 41.13.11 CPU1 ROM table register map and reset values

Table 279. CPU1 ROM table register map and reset values

Offset	Register name	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	C1ROM_MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	SYSMEM	0	
	Reset value																																	
0xFD0	C1ROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	4KCOUNT [3:0]	JEP106CON [3:0]	
	Reset value																																	
0xFE0	C1ROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	PARTNUM[7:0]	0	
	Reset value																																	
0xFE4	C1ROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	JEP106ID [3:0]	PARTNUM [11:8]	
	Reset value																																	
0xFE8	C1ROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	REVISION [3:0]	JEP106ID [6:4]	
	Reset value																																	
0xFEC	C1ROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	REVAND[3:0]	CMOD[3:0]	
	Reset value																																	
0xFF0	C1ROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	PREAMBLE[7:0]	0	
	Reset value																																	
0xFF4	C1ROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	CLASS[3:0]	PREAMBLE [11:8]	
	Reset value																																	
0xFF8	C1ROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	PREAMBLE[19:12]	0	
	Reset value																																	
0xFFC	C1ROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	1	PREAMBLE[27:20]	0	
	Reset value																																	

Refer to [Section 41.13: CPU1 ROM table](#) for the register boundary addresses.

## 41.14 CPU1 data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as:

- Watchpoint
- ETM trigger, only available on STM32WB55xx
- PC sampling trigger
- Data address sampling trigger
- Data comparator (comparator 1 only)
- Clock cycle counter comparator (comparator 0 only)

It also contains counters for:

- Clock cycles
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction
- Interrupt overhead

A DWT comparator compares the value held in its DWT\_COMPR register with one of the following:

- a data address
- an instruction address
- a data value
- the cycle count value, for comparator 0 only.

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- One or more DWT Data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value.
- A watchpoint debug event, on either the PC value or the accessed data address.
- A CMPMATCH[N] event, that signals the match outside the DWT unit.

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v7-M Architecture Reference Manual [\[5\]](#).

### 41.14.1 DWT control register (DWT\_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			NUMCOMP[3:0]	NOTR CPKT	NOEXT TRIG	NOCYC CNT	NOPRFCNT	Res.	CYCEV TENA	FOLDE VTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEV TENA	EXCTRCENA
r	r	r	r	r	r	r	r		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAMPL ENA	SYNCTAP[1:0]	CYCTAP		POSTINIT[3:0]		POSTRESET[3:0]					CYCCNT ENA	
			rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]**: Number of comparators implemented (read only)

0x4: Four comparators

Bit 27 **NOTRCPKT**: Trace sampling and exception tracing support (read only)

0x0: Supported

Bit 26 **NOEXTTRIG**: External match signal, CMPMATCH support (read only)

0x0: Supported

Bit 25 **NOCYCCNT**: Cycle counter support (read only)

0x0: Supported

Bit 24 **NOPRFCNT**: Profiling counter support (read only)

0x0: Supported

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CYCEVTENA**: Enable for POSTCNT underflow event counter packet generation

0x0: Disabled

0x1: Enabled

Bit 21 **FOLDEVTENA**: Enable for folded instruction counter overflow event generation

0x0: Disabled

0x1: Enabled

Bit 20 **LSUEVTENA**: Enable for LSU counter overflow event generation

0x0: Disabled

0x1: Enabled

Bit 19 **SLEEPEVTENA**: Enable for sleep counter overflow event generation

0x0: Disabled

0x1: Enabled

Bit 18 **EXCEVTENA**: Enable for exception overhead counter overflow event generation

0x0: Disabled

0x1: Enabled

Bit 17 **CPIEVTEVA**: Enable for CPI counter overflow event generation

0x0: Disabled

0x1: Enabled

Bit 16 **EXCTRCENA**: Enable for exception trace generation

0x0: Disabled

0x1: Enabled

Bits 15:13 Reserved, must be kept at reset value.

- Bit 12 **PCSAMPLENA**: Enables use of POSTCNT counter as a timer for Periodic PC sample packet generation.  
 0x0: Disabled  
 0x1: Enabled
- Bits 11:10 **SYNCTAP[1:0]**: Selects the position of the synchronization packet counter tap on the CYCCNT counter. This determines the synchronization packet rate.  
 0x0: Disabled. No synchronization packets  
 0x1: Tap at CYCCNT[24]  
 0x2: Tap at CYCCNT[26]  
 0x3: Tap at CYCCNT[28]
- Bit 9 **CYCTAP**: Selects the position of the POSTCNT tap on the CYCCNT counter.  
 0x0: Tap at CYCCNT[6]  
 0x1: Tap at CYCCNT[10]
- Bits 8:5 **POSTINIT[3:0]**: Initial value of the POSTCNT counter. Writes to this field are ignored if POSTCNT counter is enabled (ie. CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).
- Bits 4:1 **POSTPRESET[3:0]**: Reload value of the POSTCNT counter.
- Bit 0 **CYCCNTENA**: Enables CYCCNT counter.  
 0x0: Disabled  
 0x1: Enabled

#### 41.14.2 DWT cycle count register (DWT\_CYCCNTR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCCNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CYCCNT[31:0]**: Processor clock cycle counter

#### 41.14.3 DWT CPI count register (DWT\_CPICNTR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	rw						
CPICNT[7:0]															

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]**: CPI counter. Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT\_LSUCNTR, and counts any instruction fetch stalls.

#### 41.14.4 DWT exception count register (DWT\_EXCCNTR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	rw						
EXCCNT[7:0]															

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: Exception overhead cycle counter. Counts the number of cycles spent in exception processing.

#### 41.14.5 DWT sleep count register (DWT\_SLPCNTR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	rw						
SLEEPCNT[7:0]															

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEPCNT[7:0]**: Sleep cycle counter. Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

#### 41.14.6 DWT LSU count register (DWT\_LSUCNTR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								LSUCNT[7:0]							
								rw							

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]**: Load store counter. Counts additional cycles required to execute load and store instructions.

#### 41.14.7 DWT fold count register (DWT\_FOLDCNTR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FOLDCNT[7:0]														
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]**: Folded instruction counter. Increments on each instruction that takes 0 cycles.

#### 41.14.8 DWT program counter sample register (DWT\_PCSR)

Address offset: 0x01C

Reset value: 0x0000 0000

Bits 31:0 **EIASAMPLE[31:0]**: Executed instruction address sample value. Samples the current value of the program counter.

#### 41.14.9 DWT comparator register x (DWT\_COMPxR)

Address offset:  $0x020 + x * 0x10$  (for  $x = 0$  to  $3$ )

Reset value: 0x0000 0000

Bits 31:0 **COMP[31:0]**: Reference value for comparison.

#### 41.14.10 DWT mask register x (DWT\_MASKxR)

Address offset:  $0x024 + x * 0x10$  (for  $x = 0$  to  $3$ )

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
MASK[4:0]															
												rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **MASK[4:0]**: Comparator mask size. Provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

#### 41.14.11 DWT function register x (DWT\_FUNCTxR)

Address offset: 0x028 + x \* 0x10 (for x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]
															rw rw rw rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAVADDR0[3:0]				DATAVSIZE[1:0]		LINK1 ENA	DATAV MATCH	CYC MATCH	Res.	EMIT RANGE	Res.	FUNCTION[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match (read only). Indicates if a comparator match has occurred since the register was last read.

0: No match

1: Match occurred

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **DATAVADDR1[3:0]**: When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a second comparator to use for linked address comparison.

Bits 15:12 **DATAVADDR0[3:0]**: When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a comparator to use for linked address comparison.

Bits 11:10 **DATAVSIZE[1:0]**: For data value matching, specifies the size of the required data comparison.

0x0: Byte

0x1: Half word

0x2: Word

0x3: Reserved

Bit 9 **LNK1ENA**: Indicates whether use of a second linked comparator is supported (read only).

0x1: Supported

Bit 8 **DATAVMATCH**: Enables cycle comparison.

0x0: Perform address comparison

0x1: Perform data value comparison

Bit 7 **CYCMATCH**: Enables cycle count comparison on comparator 0. This field is reserved for other comparators.

0x0: No cycle count comparison

0x1: Compare DWT\_COMP0R with the cycle counter, DWT\_CYCCNTR

Bit 6 Reserved, must be kept at reset value.

Bit 5 **EMITRANGE**: Enables generation of data trace address offset packets (containing data address bits 0 to 15)

0x0: Disabled

0x1: Enabled

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **FUNCTION[3:0]**: Selects action to take on comparator match. The meaning of this bit field depends on the setting of the DATAVMATCH and CYCMATCH fields. See [\[5\]](#).

#### 41.14.12 DWT CoreSight peripheral identity register 4 (DWT\_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size

0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x4: Arm® JEDEC code

#### 41.14.13 DWT CoreSight peripheral identity register 0 (DWT\_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]

0x02: DWT part number

#### 41.14.14 DWT CoreSight peripheral identity register 1 (DWT\_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]  
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]  
0x9: DWT part number

#### 41.14.15 DWT CoreSight peripheral identity register 2 (DWT\_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]									
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bit 7:4 **REVISION[3:0]**: Component revision number  
0x3: r0p4

Bit 3 **JEDEC**: JEDEC assigned value  
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]  
0x3: Arm® JEDEC code

#### 41.14.16 DWT CoreSight peripheral identity register 3 (DWT\_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVAND[3:0]				CMOD[3:0]										
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified  
0x0: No customer modifications

#### 41.14.17 DWT CoreSight component identity register 0 (DWT\_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]

0x0D: Common ID value

#### 41.14.18 DWT CoreSight peripheral identity register 1 (DWT\_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[11:8]							
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class

0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]

0x0: Common ID value

#### 41.14.19 DWT CoreSight component identity register 2 (DWT\_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]

0x05: Common ID value

#### 41.14.20 DWT CoreSight component identity register 3 (DWT\_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]

0xB1: Common ID value

#### 41.14.21 CPU1 DWT register map and reset values

The CPU1 DWT registers are located at address range 0xE0001000 to 0xE0001FFC.

**Table 280. CPU1 DWT register map and reset values**

**Table 280. CPU1 DWT register map and reset values (continued)**

Refer to [Section 41.13: CPU1 ROM table](#) for the register boundary addresses.

## 41.15 CPU1 instrumentation trace macrocell (ITM)

The ITM generates trace information as packets four sources can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

1. Software trace

Software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

2. Hardware trace

The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.

3. Local timestamping

The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

### 41.15.1 ITM stimulus register x (ITM\_STIMRx)

Address offset: 0x000 + x \* 0x4 (x = 0 to 31)

Reset value: Unknown

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMULUS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMULUS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **STIMULUS[31:0]**: Write data is output on the trace bus as a software event packet. When reading, bit 0 is a FIFOREADY indicator:

0: Stimulus port buffer is full (or port is disabled)

1: Stimulus port can accept new write data

### 41.15.2 ITM trace enable register (ITM\_TER)

Address offset: 0x080

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMENA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMENA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **STIMENA[31:0]**: Each bit n (0:31) enables the stimulus port associated with the `ITM_STIMRn` register.

0: Port disabled

1: Port enabled

#### 41.15.3 ITM trace privilege register (ITM\_TPR)

Address offset: 0xE00

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIVMASK[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIVSK[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PRIVMASK[31:0]**: Enable unprivileged access to ITM stimulus ports. Each bit controls eight stimulus ports.

0bXXX0: Unprivileged access permitted on ports 0 to 7

0bXXX1: Only privileged access permitted on ports 0 to 7

0bXX0X: Unprivileged access permitted on ports 8 to 15

0bXX1X: Only privileged access permitted on ports 8 to 15

0bX0XX: Unprivileged access permitted on ports 16 to 23

0bX1XX: Only privileged access permitted on ports 16 to 23

0b0XXX: Unprivileged access permitted on ports 24 to 31

0b1XXX: Only privileged access permitted on ports 24 to 31

#### 41.15.4 ITM trace control register (ITM\_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BUSY	TRACEBUSID[6:0]												
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE[1:0]	Res.	Res.	Res.	SWOENA	TXENA	SYNCENA	TSENA	ITMENA	
						rw	rw			r	rw	rw	rw	rw	

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY**: Indicates whether the ITM is currently processing events (read only).

0: Not busy

1: Busy

Bits 22:16 **TRACEBUSID[6:0]**: Identifier for multi-source trace stream formatting. If multi-source trace is in use, the debugger must write a non-zero value to this field. Note: different IDs must be used for each trace source in the system.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TSPRESCALE[1:0]**: Local timestamp prescaler, used with the trace packet reference clock.

The possible values are:

0x0: No prescaling.

0x1: Divide by 4.

0x2: Divide by 16.

0x3: Divide by 64.

Bit 7:5 Reserved, must be kept at reset value.

Bit 4 **SWOENA**: Enables asynchronous clocking of the timestamp counter (read only).

0: Timestamp counter uses processor clock

Bit 3 **TXENA**: Enables forwarding of hardware event packets from the DWT unit to the trace port.

0: Disabled

1: Enabled

Bit 2 **SYNCENA**: Enables synchronization packet transmission.

*Note: The debugger setting this bit must also configure the DWT\_CTRLR register SYNCTAP field in the DWT for correct synchronization speed.*

0: Disabled

1: Enabled

Bit 1 **TSENA**: Enables local timestamp generation.

0: Disabled

1: Enabled

Bit 0 **ITMENA**: Enables the ITM.

0: Disabled

1: Enabled

#### 41.15.5 ITM CoreSight peripheral identity register 4 (ITM\_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		4KCOUNT[3:0]		JEP106CON[3:0]											
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size

0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x4: Arm® JEDEC code

#### 41.15.6 ITM CoreSight peripheral identity register 0 (ITM\_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		PARTNUM[7:0]													
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]

0x01: ITM part number

#### 41.15.7 ITM CoreSight peripheral identity register 1 (ITM\_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		JEP106ID[3:0]		PARTNUM[11:8]											
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]  
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]  
0x0: ITM part number

#### 41.15.8 ITM CoreSight peripheral identity register 2 (ITM\_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]									
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bit 7:4 **REVISION[3:0]**: Component revision number  
0x3: r0p4

Bit 3 **JEDEC**: JEDEC assigned value  
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]  
0x3: Arm® JEDEC code

#### 41.15.9 ITM CoreSight peripheral identity register 3 (ITM\_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVAND[3:0]				CMOD[3:0]										
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified  
0x0: No customer modifications

#### 41.15.10 ITM CoreSight component identity register 0 (ITM\_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.							PREAMBLE[7:0]								
									r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]

0xD: Common ID value

#### 41.15.11 ITM CoreSight peripheral identity register 1 (ITM\_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.							PREAMBLE[11:8]								
									r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class

0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]

0x0: Common ID value

#### 41.15.12 ITM CoreSight component identity register 2 (ITM\_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]

0x05: Common ID value

#### 41.15.13 ITM CoreSight component identity register 3 (ITM\_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]

0xB1: Common ID value

#### 41.15.14 ITM register map and reset values

The ITM registers are located at address range 0xE0000000 to 0xE0000FFC.

Table 281. CPU1 ITM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 to 0x07C	ITM_STIM0-31R																																
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
0x0E00	ITM_TER																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0E40	ITM_TPR																																
	Reset value																																
0xE80	ITM_TCR																																
	Reset value																																
0xFD0	ITM_PIDR4																																
	Reset value																																
0xFE0	ITM_PIDR0																																
	Reset value																																
0xFE4	ITM_PIDR1																																
	Reset value																																
0xFE8	ITM_PIDR2																																
	Reset value																																
0xFEC	ITM_PIDR3																																
	Reset value																																
0xFF0	ITM_CIDR0																																
	Reset value																																
0xFF4	ITM_CIDR1																																
	Reset value																																
0xFF8	ITM_CIDR2																																
	Reset value																																
0xFFC	ITM_CIDR3																																
	Reset value																																

Refer to [Section 41.13: CPU1 ROM table](#) for the register boundary addresses.

## 41.16 CPU1 breakpoint unit (FPB)

The FPB allows the user to set hardware breakpoints. It contains six comparators that monitor the instruction fetch address and two literal address comparators. If a match occurs, the address is remapped to an address in system memory, defined by the FPB\_REMAPR register plus an offset corresponding to the matching comparator. Alternatively, the instruction comparators can be configured to generate a breakpoint instruction.

### 41.16.1 FPB control register (FPB\_CTRLR)

Address offset: 0x000

Reset value: 0x0000 0260

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE[6:4]			NUM_LIT[3:0]				NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
r	r	r	r	r	r	r	r	r	r	r	r			rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **NUM\_CODE[6:4]**: Number of instruction address comparators supported - least significant bits (read only).

0x0: 6 instruction comparators supported.

Bits 11:8 **NUM\_LIT[3:0]**: Number of literal address comparators supported (read only).

0x2: Two literal comparators supported.

Bits 7:4 **NUM\_CODE[3:0]**: Number of instruction address comparators supported - least significant bits (read only).

0x6: 6 instruction comparators supported

Bit 1 **KEY**: Write protect key. A write to FPB\_CTRLR register is ignored if this bit is not set to 1.

Bits 0 **ENABLE**: FPB enable

0x0: Disable

0x1: Enable

### 41.16.2 FPB remap register (FPB\_REMAPR)

Address offset: 0x004

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	RMPSPT	REMAP[23:11]												
		r	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
REMAP[10:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **RMPSPt**: Indicates whether Flash memory patch remap is supported (read only).  
0x1: Remapping supported.

Bits 28:5 **REMAP[23:0]**: Remap target address. Bits [28:5] of the base address in SRAM to which the FPB remaps the address. The remap base address must be aligned to the number of words required to support the implemented comparators, that is to (NUM\_CODE+NUM\_LIT) words, with a minimum alignment of 8 words. Because remap is into the SRAM memory region, 0x20000000-0x3FFFFFF, bits [31:29] of the remap address are 0b001.

Bits 4:0 Reserved, must be kept at reset value.

### 41.16.3 FPB comparator registers (FPB\_COMPxR)

Address offset: 0x008 + x \* 0x4 (for x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REPLACE[1:0]	Res.														
rw			rw	rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														Res.	ENABLE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:30 **REPLACE[1:0]**: Defines the behavior when a match occurs between the COMP field and the instruction fetch address.

- 0x0: Reserved
- 0x1: Breakpoint on lower half-word, upper half-word is unaffected.
- 0x2: Breakpoint on upper half-word, lower half-word is unaffected.
- 0x3: Breakpoint on both upper and lower half-words.

Bit 29 Reserved, must be kept at reset value.

Bits 28:2 **COMP[26:0]**: Value to compare with address bits 28:2 of accesses to instruction code memory (0x00000000 to 0x1FFFFFF). If a match occurs, the action to be taken is defined by the REPLACE field.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Comparator enable. The comparator is only enabled if both this bit and the FPB\_ENABLE bit in the FPB\_CTRLR register are set.

- 0: Disabled
- 1: Enabled

### 41.16.4 FPB CoreSight peripheral identity register 4 (FPB\_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		4KCOUNT[3:0]		JEP106CON[3:0]											
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size

0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x4: Arm® JEDEC code

#### 41.16.5 FPB CoreSight peripheral identity register 0 (FPB\_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		PARTNUM[7:0]													
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]

0x03: FPB part number

#### 41.16.6 FPB CoreSight peripheral identity register 1 (FPB\_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		JEP106ID[3:0]		PARTNUM[11:8]											
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]  
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]  
0x0: FPB part number

#### 41.16.7 FPB CoreSight peripheral identity register 2 (FPB\_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]									
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bit 7:4 **REVISION[3:0]**: Component revision number  
0x2: r0p3

Bit 3 **JEDEC**: JEDEC assigned value  
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]  
0x3: Arm® JEDEC code

#### 41.16.8 FPB CoreSight peripheral identity register 3 (FPB\_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVAND[3:0]				CMOD[3:0]										
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified  
0x0: No customer modifications

### 41.16.9 FPB CoreSight component identity register 0 (FPB\_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.							PREAMBLE[7:0]								
									r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]

0x0D: Common ID value

### 41.16.10 FPB CoreSight peripheral identity register 1 (FPB\_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.							PREAMBLE[11:8]								
									r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class

0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]

0x0: Common ID value

### 41.16.11 FPB CoreSight component identity register 2 (FPB\_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]

0x05: Common ID value

#### 41.16.12 FPB CoreSight component identity register 3 (FPB\_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]

0xB1: Common ID value

### 41.16.13 FPB register map and reset values

The CPU1 FPB registers are located at address range 0xE0002000 to 0xE0002FFC.

**Table 282. CPU1 FPB register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
0x000	FPB_CTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x004	FPB_REMAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x008 to 0x024	FPB_COMP0-7R	REPLACE[1:0]	Res.																																				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFD0	FPB_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFE0	FPB_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFE4	FPB_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFE8	FPB_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFEC	FPB_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFF0	FPB_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFF4	FPB_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFF8	FPB_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFFC	FPB_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 41.13: CPU1 ROM table](#) for the register boundary addresses.

## 41.17 CPU1 Embedded trace macrocell (ETM™), only available on STM32WB55xx

The CPU1 ETM™ is a CoreSight™ component closely coupled to the CPU. The ETM™ generates trace packets that enable to trace the execution of the CPU1 core. It is configured for instruction trace only, i.e. data accesses are not included in the trace information.

The ETM™ receives information from the CPU over the processor trace interface, including:

- The number of instructions executed in the same cycle
- Changes in program flow
- The current processor instruction state
- The addresses of memory locations accessed by load and store instructions
- The type, direction and size of a transfer
- Condition code information
- Exception information
- Wait for interrupt state information

For more information, refer to the Arm® CoreSight™ ETM™-Cortex®-M4 technical reference manual.

### 41.17.1 ETM control register (ETM\_CR)

Address offset: 0x000

Reset value: 0x0000 0411

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TSEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ETMEN	PROG	DBRQ	BO	STALL	Res.	Res.	Res.	Res.	Res.	Res.	PDN
				rw	rw	rw	rw	rw							rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TSEN**: Timestamp Enable.

- 0: Timestamping disabled
- 1: Timestamping enabled

Bits 26:12 Reserved, must be kept at reset value.

Bit 11 **ETMEN**: ETM enable.

- 0: Trace output disabled
- 1: Trace output enabled

Bit 10 **PROG**: ETM programming. This bit must be set to 1 before programming the ETM™. Tracing is prevented while this bit is set to 1.

- 0x0: ETM operational
- 0x1: ETM in programming state

Bit 9 **BO**: Branch output. When set to 1 all branch addresses are output, even if the branch was because of a direct branch instruction. Setting this bit enables reconstruction of the program flow without having access to the memory image of the code being executed.

When this bit is set to 1, more trace data is generated, and this may affect the performance of the trace system. Information about the execution of a branch is traced regardless of the state of this bit.

Bit 8 **STALL**: Stall processor. The FIFOFULL output can be used to stall the processor to prevent overflow. The FIFOFULL output is only enabled when the stall processor bit is set to 1. When the bit is 0 the FIFOFULL output remains LOW at all times and the FIFO overflows if there are too many trace packets. Trace resumes without corruption once the FIFO has drained, if overflow does occur.

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **PDN**: ETM power down. This bit can be used by an implementation to control if the ETM is in a low power state. This bit must be cleared by the trace software tools at the beginning of a debug session.

When this bit is set to 1, writes to some registers and fields might be ignored. You can always write to the following registers and fields:

- ETMCR bit [0]
- ETMLAR
- ETMCLAIMSET register
- ETMCLAIMCLR register

When the ETMCR is written with this bit set to 1, bits other than bit [0] might be ignored.

## 41.17.2 ETM configuration code register (ETM\_CCR)

Address offset: 0x004

Reset value: 0x8C80 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDPRES	Res.	Res.	Res.	CPMAC	TSPRES	CIDCMP[1:0]	FFPRES	NEXTI[2:0].				NEXTI[2:0].			
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
NCNT[2:0].			NMMDEC[4:0].				NDVCMP[3:0]				NADCMP[3:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 30 **IDPRES**: ETM ID register present.

1: ETMIDR register is present and defines the ETM architecture version in use.

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 **CPMAC**: Co-processor and memory access.

1: Memory-mapped access to registers is supported.

Bit 26 **TSPRES**: Trace start/stop block present.

1: Trace start/stop block is present.

Bits 25:24 **CIDCMP[1:0]**: Number of context ID comparators.

0x0: Context ID comparators are not implemented.

- Bit 23 **FFPRES**: FIFOFULL logic present. To use FIFOFULL the system must also support the function, as indicated by bit[8] of ETMSCR.  
1: FIFOFULL logic is present in the ETM.
- Bits 22:20 **NEXTO[2:0]**: Number of external outputs.  
0x0: No external outputs are supported
- Bits 19:17 **NEXTI[2:0]**: Number of external inputs.  
0x2: Two external inputs are supported
- Bit 16 **SEQPRES**: Sequencer present.  
0: The sequencer is not present in the ETM.
- Bits 15:13 **NCNT[2:0]**: Number of counters.  
0x1: One counter is implemented
- Bits 12:8 **NMMDEC[4:0]**: Number of memory map decoders.  
0x0: No memory map decoder inputs are implemented.
- Bits 7:4 **NDVCM[3:0]**: Number of data value comparators.  
0x0: No data value comparators are implemented.
- Bits 3:0 **NADCMP[3:0]**: Number of address comparator pairs.  
0x0: No address comparator pairs are implemented.

### 41.17.3 ETM trigger register (ETM\_TRIGGER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCN [1:0]		RESOURCEB[6:0]										RESOURCEA[6:0]			
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.

Bits 16:14 **FCN[2:0]**: Boolean function. If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

- 0x0: A
- 0x1: NOT A
- 0x2: A AND B
- 0x3: NOT A AND B
- 0x4: NOT A AND NOT B
- 0x5: A OR B
- 0x6: NOT A OR B
- 0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier. See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier. Bits [6:4] define the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.

**Type (bits [6:4]) Index (bits [3:0])**

0x2	0-3	Embedded-ICE watch point comparators 1-4 (from DWT)
0x4	0	Counter 1 at zero
0x5	15	Trace start/stop resource
0x6	0-1	External inputs 1-2
	15	Hard-wired input, always true

#### 41.17.4 ETM status register (ETM\_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TRIGFL	TSSRSTAT	PROGVAL	UOVFL											
												rw	rw	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **TRIGFL**: Trigger flag. Set when the trigger occurs, and prevents the trigger from being output until the ETM is programmed again.

Bit 2 **TSSRSTAT**: Trigger start/stop resource status. Holds the current status of the trace start/stop resource. If set to 1, it indicates that a trace on address has been matched, without a corresponding trace off address match.

Bit 1 **PROGVAL**: Programming bit value (read only). The current effective value of the ETM Programming bit, bit [10] of the ETM\_CR. You must set the programming bit to 1 and wait for this bit to go to 1 before you start to program the ETM.

This bit remains at 0 until the FIFO is empty, or if OS lock is set in the ETM\_OSLSR register.

Bit 0 **UOVFL**: Untraced overflow (read only). If set to 1, there is an overflow that has not yet been traced. This bit is cleared to 0 when trace is restarted, or when the ETM power down bit (bit [0] of the ETM\_CR register) is set to 1.

#### 41.17.5 ETM status register (ETM\_SCR)

Address offset: 0x014

Reset value: 0x0002 0D09

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOFTCH COMP	Res.
														r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NSUPPROC[2:0]			PORT MODESUP	PORT SIZESUP	MAXPORT SIZE[3]	FFSUP	Res.	Res.	Res.	Res.	Res.	MAXPORTSIZE[2:0]		
	r	r	r	r	r	r	r						r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **NOFTCHCOMP**: No fetch comparisons.

1: Fetch comparisons are not implemented

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:12 **NSUPPROC[2:0]**: Number of supported processors.

0x0: One processor supported.

Bit 11 **PORTMODESUP**: Port mode support.

0: Current selected port mode is not supported.

1: Current selected port mode is supported.

Bit 10 **PORTSIZESUP**: Port size support.

0: Current selected port size is not supported.

1: Current selected port size is supported.

Bit 9 **MAXPORTSIZE[3]**: Maximum ETM port size bit [3]. This bit is used in conjunction with bits [2:0]

Bit 8 **FFSUP**: FIFOFULL support.

1: FIFOFULL is supported.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **MAXPORTSIZE[2:0]**: Maximum ETM port size bit [2:0]. These bits in conjunction with bit 9 indicate the maximum ETM port size supported.

0x1: Maximum port size = 1

#### 41.17.6 ETM trace enable event register (ETM\_TEEVR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCN [1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
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.

Bits 16:14 **FCN[2:0]**: Boolean function. If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

- 0x0: A
- 0x1: NOT A
- 0x2: A AND B
- 0x3: NOT A AND B
- 0x4: NOT A AND NOT B
- 0x5: A OR B
- 0x6: NOT A OR B
- 0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier. See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier. Bits [6:4] define the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.

**Type (bits [6:4]) Index (bits [3:0])**

0x2	0-3	Embedded-ICE watch point comparators 1-4 (from DWT).
0x4	0	Counter 1 at zero
0x5	15	Trace start/stop resource
0x6	0-1 15	External inputs 1-2 Hard-wired input, always true

#### 41.17.7 ETM trace enable control 1 register (ETM\_TECR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	ENONOFF	Res.								
							rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **ENONOFF**: Trace start/stop trace enable control.

- 0: Trace start/stop block does not control trace enable.
- 1: Trace enable is controlled by the trace start/stop block.

Bits 24:0 Reserved, must be kept at reset value.

#### 41.17.8 ETM FIFOFULL level register (ETM\_FFLR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
LEVEL[7:0]															

Bits 31:8 Reserved, must be kept at reset value.

Bit7:0 **LEVEL[7:0]**: The number of bytes left in the FIFO, below which the FIFOFULL signal is asserted. For example, setting this value to 15 causes processor stalling, if enabled, when there are less than 15 free bytes in the FIFO.

#### 41.17.9 ETM counter reload value 1 register (ETM\_CNTRLDVR1)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INICNT[15:0]															
<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>

Bits 31:16 Reserved, must be kept at reset value.

Bit15:0 **INICNT[15:0]**: Initial count. Specifies the counter reload value.

#### 41.17.10 ETM synchronization frequency register (ETM\_SYNCFR)

Address offset: 0x1E0

Reset value: 0x0000 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.												
				<b>r</b>											
FREQUENCY[11:0]															

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **FREQUENCY[11:0]**: This value is the time in bytes between synchronization points in the trace (the tools can start decompressing only at synchronization points).

0x400: The ETM uses a fixed synchronization packet generation frequency of every 1024 bytes of trace.

### 41.17.11 ETM ID register (ETM\_IDR)

Address offset: 0x1E4

Reset value: 0x4114 F250

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DESIGNER[7:0]								Res.	Res.	Res.	ABPE	SXSUPP	T32SUPP	Res.	LDPCF
r	r									r	r	r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PROCFAM[3:0]				ETMARCHMAJ[3:0]				ETMARCHMIN[3:0]				REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DESIGNER[7:0]**: Implementer code.

0x41: ARM<sup>®</sup>

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **ABPE**: Alternative ranch packet encoding.

1: ABPE is implemented.

Bit 19 **SXSUPP**: Security extensions support.

0: ETM behaves as if the processor is in secure mode at all times.

Bit 18 **T32SUPP**: 32-bit Thumb instruction tracing.

1: 32-bit Thumb instructions are traced as single instructions.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **LDPCF**: Load PC first.

0: Data tracing is not supported

Bits 15:12 **PROCFAM[3:0]**: Processor family.

0xF: Processor family is not identified in this register

Bits 11:8 **ETMARCHMAJ[3:0]**: Major ETM architecture version.

0x2: Version 3

Bits 7:4 **ETMARCHMIN[3:0]**: Minor ETM architecture version.

0x5: Version 5

Bits 3:0 **REVISION[3:0]**: Implementation revision.

0x0: Revision 0

### 41.17.12 ETM configuration code extension register (ETM\_CCER)

Address offset: 0x1E8

Reset value: 0x1854 1800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TSSIZE	TSENC	RFC	Res.	Res.	Res.	Res.	TSIMP	EIIMP	TSSUWP	NUMWPIN[3:0]			
		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
NUMIR[2:0]			SUPPDAC	RR	XXINBUS[7:0]								XXINSEL[2:0]		
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **TSSIZE**: Timestamp size.

0: 48 bits.

Bit 28 **TSENC**: Timestamp encoding.

1: Timestamp is encoded as a natural binary number.

Bit 27 **RFC**: Reduced function counter.

1: Counter is a reduced function counter.

Bits 26:23 Reserved, must be kept at reset value.

Bit 22 **TSIMP**: Timestamping implemented.

1: Timestamping implemented

Bit 21 **EIIMP**: EmbeddedICE behavior control implemented.

0: Not implemented

Bit 20 **TSSUWP**: Trace start/stop uses Embedded ICE watchpoint inputs.

1: Yes

Bits 19:16 **NUMWPIN[3:0]**: Embedded ICE watchpoint inputs. Number of inputs coming from the DWT.

0x4: Four inputs

Bits 15:13 **NUMIR[2:0]**: Instrumentation resources.

0x0: None

Bit 12 **SUPPDAC**: Data address comparison support.

1: Not supported

Bit 11 **RR**: Readable registers.

1: All registers are readable

Bits 10:3 **XXINBUS[7:0]**: Extended external input bus.

0x0: Not implemented

Bits 2:0 **XXINSEL[2:0]**: Extended external input selectors.

0x0: Not implemented

#### 41.17.13 ETM trace enable start/stop EmbeddedICE control register (ETM\_TESSEICR)

Address offset: 0x1F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	STOPRS[3:0]														
															rw rw rw rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	STARTRS[3:0]														
															rw rw rw rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **STOPRS[3:0]**: Stop resource selection. Setting any of these bits to 1 selects the corresponding EmbeddedICE watchpoint input as a TraceEnable stop resource. Bit [16] corresponds to input 1, bit [17] corresponds to input 2, bit [18] corresponds to input 3, and bit [19] corresponds to input 4.

Bits 3:0 **STARTRS[3:0]**: Start resource selection. Setting any of these bits to 1 selects the corresponding EmbeddedICE watchpoint input as a TraceEnable start resource. Bit [0] corresponds to input 1, bit [1] corresponds to input 2, bit [2] corresponds to input 3, and bit [3] corresponds to input 4.

#### 41.17.14 ETM timestamp event register (ETM\_TSEVR)

Address offset: 0x1F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCN [1:0]		RESOURCEB[6:0]								RESOURCEA[6:0]					
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.

Bits 16:14 **FCN[2:0]**: Boolean function. If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

- 0x0: A
- 0x1: NOT A
- 0x2: A AND B
- 0x3: NOT A AND B
- 0x4: NOT A AND NOT B
- 0x5: A OR B
- 0x6: NOT A OR B
- 0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier. See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier. Bits [6:4] define the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.

**Type (bits [6:4]) Index (bits [3:0])**

0x2	0-3	Embedded-ICE watchpoint comparators 1-4 (from DWT)
0x4	0	Counter 1 at zero
0x5	15	Trace start/stop resource
0x6	0-1	External inputs 1-2
	15	Hard-wired input, always true

**41.17.15 ETM trace ID register (ETM\_TRACEIDR)**

Address offset: 0x200

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
TRACEID[6:0]															
										rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace ID to output onto the trace bus. This should be programmed with a unique value to differentiate it from other trace sources in the system.**41.17.16 ETM ID register 2(ETM\_IDR2)**

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.														
														SWPTO	RFETO
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWPTO**: Identifies the order for a SWP or SWPB instruction.

0: The Load transfer is traced before the Store transfer

Bit 0 **RFETO**: Identifies the order for a RFE instruction.

0: The PC transfer is traced before the CPSR transfer

**41.17.17 ETM device power down status register 2(ETM\_PDSR)**

Address offset: 0x314

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	POWER														
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **POWER**: ETM powered up.

1: ETM trace registers can be accessed.

#### 41.17.18 ETM claim tag set register (ETM\_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CLAIMSET[3:0]														
															rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:

- 0000: No effect
- xxx1: Set bit 0
- xx1x: Set bit 1
- x1xx: Set bit 2
- 1xxx: Set bit 3

Read:

0xF: Indicates there are four bits in claim tag

#### 41.17.19 ETM claim tag clear register (ETM\_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
CLAIMCLR[3:0]																
													rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:

0000: No effect  
xxx1: Clear bit 0  
xx1x: Clear bit 1  
x1xx: Clear bit 2  
1xxx: Clear bit 3

Read: Returns current value of claim tag

#### 41.17.20 ETM lock access register (ETM\_LAR)

Address offset: 0xFB0

Reset value: N/A

Bits 31:0 **ACCESS\_W[31:0]**: Enables write access to some ETM registers by processor cores (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access

### Other values: Disable write access

#### 41.17.21 ETM lock status register (ETM LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LOCK TYPE	LOCK GRANT	LOCK EXIST												
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Indicates the size of the ETM\_LAR register  
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock. This bit always reads as zero by an external debugger.  
0: Write access is permitted  
1: Write access is blocked. Only reads are permitted.

Bit 0 **LOCKEXIST**: Indicates whether a lock control mechanism exists. This bit always reads as zero by an external debugger.  
0: No lock control mechanism exists.  
1: Lock control mechanism is implemented

#### 41.17.22 ETM authentication status register (ETM\_AUTHSTATR)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SNID[1:0]	SID[1:0]	NSNID[1:0]	NSID[1:0]											
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug  
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug  
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug  
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug  
0x0: Not implemented

#### 41.17.23 ETM CoreSight device identity register (ETM\_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Device sub-type identifier

0x1: Processor trace

Bits 3:0 **MAJORTYPE[3:0]**: Device main type identifier

0x3: Trace source

#### 41.17.24 ETM CoreSight peripheral identity register 4 (ETM\_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	4KCOUNT[3:0]				JEP106CON[3:0]										
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size

0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x4: <sup>®</sup> JEDEC code

#### 41.17.25 ETM CoreSight peripheral identity register 0 (ETM\_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0025

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PARTNUM[7:0]														
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]

0x25: ETM part number

#### 41.17.26 ETM CoreSight peripheral identity register 1 (ETM\_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		JEP106ID[3:0]			PARTNUM[11:8]										
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]

0xB: <sup>®</sup> JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]

0x9: ETM part number

#### 41.17.27 ETM CoreSight peripheral identity register 2 (ETM\_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		REVISION[3:0]		JEDEC		JEP106ID[6:4]									
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bit 7:4 **REVISION[3:0]**: Component revision number

0x0: r0p1

Bit 3 **JEDEC**: JEDEC assigned value

0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]

0x3: Arm<sup>®</sup> JEDEC code

#### 41.17.28 ETM CoreSight peripheral identity register 3 (ETM\_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version

0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified

0x0: No customer modifications

#### 41.17.29 ETM CoreSight component identity register 0 (ETM\_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]

0x0D: Common ID value

#### 41.17.30 ETM CoreSight peripheral identity register 1 (ETM\_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class  
0x9: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]  
0x0: Common ID value

### 41.17.31 ETM CoreSight component identity register 2 (ETM\_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREAMBLE[19:12]														
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]  
0x05: Common ID value

### 41.17.32 ETM CoreSight component identity register 3 (ETM\_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREAMBLE[27:20]														
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]  
0xB1: Common ID value

### 41.17.33 ETM register map and reset values

The ETM registers are accessed by the debugger via the CPU1 AHB-AP, at address range 0xE0041000 to 0xE0041FFC.

**Table 283. CPU1 ETM register map and reset values**

Offset	Register name	Reset value
0x000	ETM_CR	31
	Reset value	0
0x004	ETM_CCR	28
	Reset value	1
0x008	ETM_TRIGGER	27
	Reset value	1
0x010	ETM_SR	26
	Reset value	0
0x014	ETM_SCR	25
	Reset value	0
0x020	ETM_TEEVR	24
	Reset value	1
0x024	ETM_TECR1	23
	Reset value	0
0x028	ETM_FFLR	22
	Reset value	1
0x140	ETM_CNTRLDVR1	21
	Reset value	0
0x1E0	ETM_SYNCFR	20
	Reset value	1
0x1E4	ETM_IDR	19
	Reset value	0

**Table 283. CPU1 ETM register map and reset values (continued)**

Table 283. CPU1 ETM register map and reset values (continued)

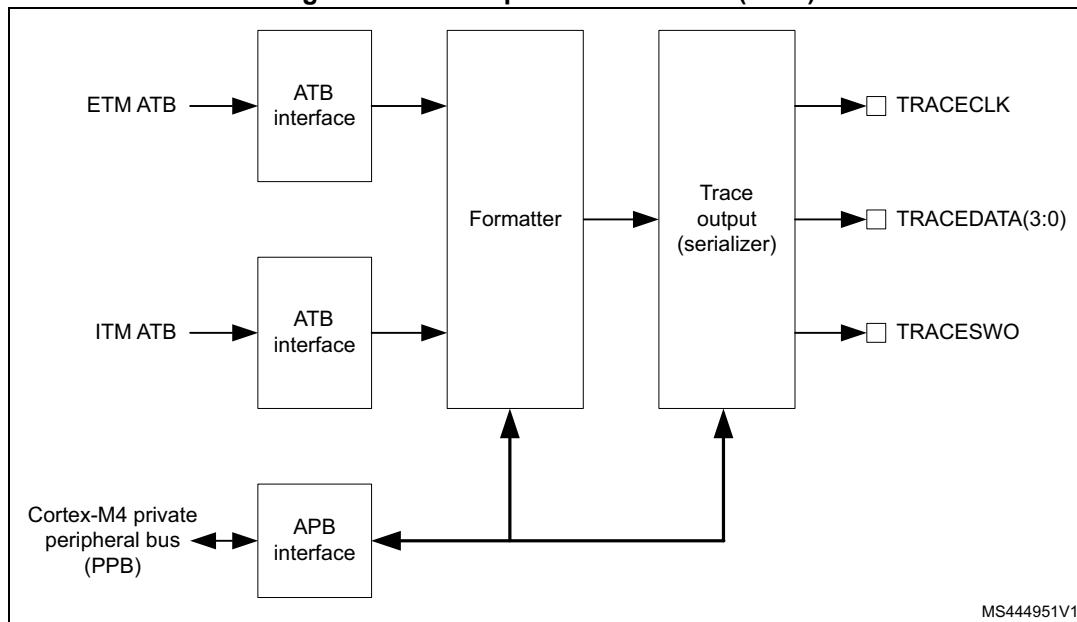
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFEC	ETM_PIDR3	Res.																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xFF0	ETM_CIDR0	Res.																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xFF4	ETM_CIDR1	Res.																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xFF8	ETM_CIDR2	Res.																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xFFC	ETM_CIDR3	Res.																															
	Reset value	1	0	1	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	

Refer to [Section 41.13: CPU1 ROM table](#) for the register boundary addresses.

## 41.18 CPU1 trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. As shown in [Figure 421](#), the TPIU has two ATB slave ports for incoming trace data from the ETM and ITM respectively. The trace port is a synchronous parallel port, comprising a clock output, TRACECLK, and four data outputs, TRACEDATA(3:0). The trace port width is programmable in the range 1 to 4. Using a smaller port width reduces the number of test points/connector pins needed, and frees up IOs for other purposes, at the expenses of bandwidth restriction of the trace port, and hence of the quantity of trace information that can be output in real time.

**Figure 421. Trace port interface unit (TPIU)**



Trace data can also be output on the serial wire output, TRACESWO.

For more information on the Trace Port Interface in the CPU1 refer to the Arm® Cortex®-M4 Technical Reference Manual [\[2\]](#).

### 41.18.1 TPIU supported port size register (TPIU\_SSPSR)

Address offset: 0x000

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: Indicates supported trace port sizes, from 1 to 32 pins. Bit n-1 when set indicates that port size n is supported.

0x0000 000F: Port sizes 1 to 4 supported

#### 41.18.2 TPIU current port size register (TPIU\_CSPSR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: Indicates current trace port size. Bit n-1 when set indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result. This register should only be modified when the formatter is stopped.

#### 41.18.3 TPIU asynchronous clock prescaler register (TPIU\_ACPR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PRESCALER[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]**: Selects the baud rate for the asynchronous output, TRACESWO. The baud rate is given by the TRACELKIN frequency divided by (PRESCALER +1).

#### 41.18.4 TPIU selected pin protocol register (TPIU\_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TXMODE[1:0]														
															rw rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TXMODE[1:0]**: Selects the protocol used for trace output.

0x0: Parallel trace port mode, only available on STM32WB55xx

0x1: Asynchronous SWO using Manchester encoding

0x2: Asynchronous SWO using NRZ encoding

0x3: Reserved

#### 41.18.5 TPIU formatter and flush status register (TPIU\_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG											
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP**: Indicates whether formatter can be stopped or not.

1: Formatter cannot be stopped

Bit 2 **TCPRESENT**: Indicates whether the optional TRACECTL output pin is available for use.

0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.

0: Formatter has not stopped

1: Formatter has stopped

Bit 0 **FLINPROG**: Flush in progress. Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU\_FFCR register.

0: No flush in progress

1: Flush in progress

#### 41.18.6 TPIU formatter and flush control register (TPIU\_FFCR)

Address offset: 0x304

Reset value: 0x0000 0102

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TRIGIN	Res.	Res.	Res.	Res.	Res.	Res.	ENFCONT	Res.						
							r							rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **TRIGIN**: Trigger on trigger in.

1: Indicate a trigger in the trace stream when the TRIGIN input is asserted.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT**: Enable continuous formatting. Setting this bit to zero in SWO mode bypasses the formatter and only ITM/DWT trace is output. ETM trace is discarded.

0: Continuous formatting is disabled

1: Continuous formatting is enabled

Bit 0 Reserved, must be kept at reset value.

#### 41.18.7 TPIU formatter synchronization counter register (TPIU\_FSCR)

Address offset: 0x308

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CYCCOUNT[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **CYCCOUNT[12:0]**: Enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word sync frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

#### 41.18.8 TPIU claim tag set register (TPIU\_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
														rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:

0000: No effect  
xxx1: Set bit 0  
xx1x: Set bit 1  
x1xx: Set bit 2  
1xxx: Set bit 3

Read:

0xF: Indicates there are four bits in claim tag

#### 41.18.9 TPIU claim tag clear register (TPIU\_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
														rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:

0000: No effect  
xxx1: Clear bit 0  
xx1x: Clear bit 1  
x1xx: Clear bit 2  
1xxx: Clear bit 3

Read: Returns current value of claim tag

#### 41.18.10 TPIU device configuration register (TPIU\_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0000 0CA1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWO UARTNRZ	SWO MAN	TCLK DATA	FIFO SIZE[2:0]			CLK RELAT	MAXNUM[3:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 11 **SWONRZ**: Indicates whether Serial wire output, NRZ, is supported.

0x1: Supported

Bit 10 **SWOMAN**: Indicates whether Serial wire output, Manchester encoded format, is supported.

0x1: Supported

Bit 9 **TCLKDATA**: Indicates whether trace clock plus data is supported

0x0: Supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2

0x2: FIFO size = 4 bytes

Bit 5 **CLKRELAT**: Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous)

0x1: Asynchronous

Bits 4:0 **MAXNUM[4:0]**: Number/type of ATB input port multiplexing

0x1: Two input ports

#### 41.18.11 TPIU device type identifier register (TPIU\_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBTYPE[3:0]			MAJORTYPE[3:0]											
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification

0x1: Trace port component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification

0x1: Trace sink component

#### 41.18.12 TPIU CoreSight peripheral identity register 4 (TPIU\_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
								r	r	r	r	r	r	r	r
4KCOUNT[3:0]								JEP106CON[3:0]							

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size

0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x4: Arm® JEDEC code

#### 41.18.13 TPIU CoreSight peripheral identity register 0 (TPIU\_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00A1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
								r	r	r	r	r	r	r	r
PARTNUM[7:0]															

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]

0xA1: CPU1 TPIU part number

#### 41.18.14 TPIU CoreSight peripheral identity register 1 (TPIU\_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
								r	r	r	r	r	r	r	r
JEP106ID[3:0]								PARTNUM[11:8]							

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]  
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]  
0x9: CPU1 TPIU part number

#### 41.18.15 TPIU CoreSight peripheral identity register 2 (TPIU\_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 004B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]									
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bit 7:4 **REVISION[3:0]**: Component revision number  
0x4: r0p5

Bit 3 **JEDEC**: JEDEC assigned value  
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]  
0x3: Arm® JEDEC code

#### 41.18.16 TPIU CoreSight peripheral identity register 3 (TPIU\_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REVAND[3:0]				CMOD[3:0]										
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified  
0x0: No customer modifications

#### 41.18.17 TPIU CoreSight component identity register 0 (TPIU\_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]

0x0D: Common ID value

#### 41.18.18 TPIU CoreSight peripheral identity register 1 (TPIU\_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PREAMBLE[11:8]							
								r	r	r	r	r	r	r	r

Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class

0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]

0x0: Common ID value

#### 41.18.19 TPIU CoreSight component identity register 2 (TPIU\_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]

0x05: Common ID value

#### 41.18.20 TPIU CoreSight component identity register 3 (TPIU\_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	r	r	r	r	r	r	r								

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]

0xB1: Common ID value

### 41.18.21 CPU1 TPIU register map and reset values

Table 284. CPU1 TPIU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x000	TPIU_SSNSR																																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x004	TPIU_CSNSR																																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x010	TPIU_ACPR	Res.																																		
	Reset value																																			
0x0F0	TPIU_SPPR																																			
	Reset value																																			
0x300	TPIU_FFSR																																			
	Reset value																																			
0x304	TPIU_FFCR																																			
	Reset value																																			
0x308	TPIU_FSCR																																			
	Reset value																																			
0xFA0	TPIU_CLAIMSETR																																			
	Reset value																																			
0xFA4	TPIU_CLAIMCLR																																			
	Reset value																																			
0xFC8	TPIU_DEVIDR																																			
	Reset value																																			
0xFD0	TPIU_DEVTYPEP	Res.																																		
	Reset value																																			
0xFD0	TPIU_PIDR4	Res.																																		
	Reset value																																			

**Table 284. CPU1 TPIU register map and reset values (continued)**

## 41.19 CPU1 cross trigger interface (CTI)

See [Section 41.6: Cross trigger interface \(CTI\) and matrix \(CTM\)](#).

## 41.20 References

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8 Aug 2013
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G, 16 March 2015
3. DDI 0461B (ID010111) - Arm® CoreSight™ Trace Memory Controller r0p1 Technical Reference Manual, Issue B, 10 Dec 2010
4. DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H, 10 July, 2009
5. DDI 0403D (ID100710) - Arm®v7-M Architecture Reference Manual, Issue E.b, 2 December 2014
6. DDI 0494-2a (ID062813) - Arm® CoreSight™ ETM™-M0+ r0p1 Technical Reference Manual, Issue D, 6 July, 2015
7. DDI 0440C (ID070610) - Arm® CoreSight™ ETM™-M4 r0p1 Technical Reference Manual, Issue C, 29 June 2012

## 42 Device electronic signature

The device electronic signature is stored in the System memory area of the Flash memory module, and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match the characteristics of the microcontroller.

### 42.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial number (USB string serial number, or other end applications)
- for use as part of the security keys, to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the memory
- to activate processes such as secure boot.

The 96-bit unique device identifier provides a reference number, unique for a given device and in any context. These bits cannot be altered by the user.

Base address: 0x1FFF 7590

Address offset: 0x00

Read only = 0xXXXX XXXX, where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID(31:16)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID(15:0)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[31:0]**: X and Y coordinates on the wafer expressed in BCD format

Address offset: 0x04

Read only = 0xXXXX XXXX, where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **UID[63:40]: LOT\_NUM[23:0]**

Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]: WAF\_NUM[7:0]**

Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX, where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]: LOT\_NUM[25:24]**

Lot number (ASCII encoded)

## 42.2 Memory size data register

### 42.2.1 Flash size data register

Base address: 0x1FFF 75E0

Address offset: 0x00

Read only = 0xXXXX, where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **FLASH\_SIZE[15:0]: Flash memory size**

Indicates the size of the device Flash memory, expressed in Kbytes.

As an example, 0x040 corresponds to 64 Kbytes.

## 42.3 Package data register

Base address: 0x1FFF 7500

Address offset: 0x00

Read only = 0xXXXX, where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	PKG[4:0]														
											r	r	r	r	r

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **PKG[4:0]**: Package type

10001: WLCSP100 / UFBGA129

10011: VFQFPN68

01010: UFQPFN48

Others: reserved

## 42.4 Part number codification register

Base address: 0x1FFF 77DC

Address offset: 0x00

Read only = 0xXXXX XXXX, where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CODIFICATION[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CODIFICATION[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CODIFICATION[31:0]**: Part number codification

0x0000 3533 STMicroelectronics STM32WB35xx part number codification

0xXXXX XXXX STMicroelectronics STM32WB55xx part number codification

## 43 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture ([www.psacertified.org](http://www.psacertified.org)) and/or Security Evaluation standard for IoT Platforms ([www.trustcb.com](http://www.trustcb.com)). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on [www.st.com](http://www.st.com) for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

## 44 Revision history

Table 285. Document revision history

Date	Revision	Changes
15-Jan-2018	1	<p>Initial release.</p>
12-Apr-2018	2	<p>Updated document title, <i>Introduction</i> and <i>Related documents</i>.            Updated <i>Figure 1: System architecture</i> and <i>Figure 2: Memory map</i>.            Updated <i>Table 1: Memory map and peripheral register boundary addresses</i>.            Updated <i>Section 2.1.4: S3: CPU2 (Cortex® -M0+)</i> <i>S-bus</i>, <i>CPU1 physical remap</i> and <i>Section 2.33: CPU2 boot</i>, and added <i>Section 2.34: CPU2 SRAM fetch disable</i>.            Updated <i>Section 3.3.4: Read access latency</i>, <i>Section 3.3.8: Flash main memory programming sequences</i>, <i>Standard programming</i>, <i>Fast programming</i>, <i>Programming errors signaled by flags</i>, <i>PGSERR</i> and <i>PGAERR</i> in a page-based row programming, <i>User and read protection option bytes</i>, <i>Secure SRAM2 start address and CPU2 reset vector option bytes</i>, <i>Option byte loading</i>, <i>Section 3.5: FLASH UID64</i>, <i>Changing the Read protection level</i>, <i>Section 3.6.4: CPU2 security (ESE)</i>, <i>CPU2 secure SRAM2 areas</i>, <i>Section 3.10.1: Flash memory access control register (FLASH_ACR)</i>, <i>Section 3.10.4: Flash memory status register (FLASH_SR)</i>, <i>Section 3.10.5: Flash memory control register (FLASH_CR)</i>, <i>Section 3.10.7: Flash memory option register (FLASH_OPTR)</i>, <i>Section 3.10.19: Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)</i> and <i>Section 3.10.16: Flash memory CPU2 status register (FLASH_C2SR)</i>.            Added <i>Programming errors causing a bus error</i> and <i>PGSERR</i> and <i>PGAERR</i> in a page-based row programming.            Updated <i>Table 4: Number of wait states vs. Flash memory clock (HCLK4) frequency</i>, <i>Table 9: Option bytes organization</i>, <i>Table 13: RDP regression from Level 1 to Level 0 and memory erase</i> and <i>Table 18: Flash interface register map and reset values</i>.            Updated <i>Section 5.4.2: CRC independent data register (CRC_IDR)</i> and <i>Section Table 20.: CRC register map and reset values</i>            Updated <i>Section 6.1: Power supplies</i>, <i>Section 6.4: Low-power modes</i>, <i>Section : Entering Low-power run mode</i>, <i>Section 6.6.3: PWR control register 3 (PWR_CR3)</i>, <i>Section 6.6.5: PWR status register 1 (PWR_SR1)</i>, <i>Section 6.6.7: PWR status clear register (PWR_SCR)</i>, <i>Section 6.6.8: PWR control register 5 (PWR_CR5)</i>, <i>Section 6.6.21: PWR CPU2 control register 1 (PWR_C2CR1)</i> and <i>Section 6.6.22: PWR CPU2 control register 3 (PWR_C2CR3)</i>.            Updated <i>Table 24: Low-power mode summary</i>, <i>Table 30: Stop0 mode</i>, <i>Table 31: Stop1 mode</i>, <i>Table 32: Stop2 mode</i>, <i>Table 33: Standby mode</i>, <i>Table 34: Shutdown mode</i> and <i>Table 35: PWR register map and reset values</i>.            Updated <i>Figure 7: Power supply overview</i> and <i>Figure 8: Supply configurations</i>.            Updated <i>Table 37: Peripherals interconnect matrix</i> and <i>Section 7.4.1: From timer (TIM1/TIM2/TIM17) to timer (TIM1/TIM2)</i>.</p>

Table 285. Document revision history (continued)

Date	Revision	Changes
12-Apr-2018	2 (cont'd)	<p>Updated <a href="#">Figure 15: Clock tree</a>.</p> <p>Updated <a href="#">Table 38: Maximum clock source frequency</a>, <a href="#">Table 39: SMPS step-down converter clock source selection and division</a> and <a href="#">Table 42: RCC register map and reset values</a>.</p> <p>Updated <a href="#">Section 8.2.5: PLLs</a>, <a href="#">Section 8.2.8: LSI2 clock</a>, <a href="#">Section 8.2.9: System clock (SYSCLK) selection</a>, <a href="#">Section 8.2.13: LSI source selection</a>, <a href="#">Section 8.2.20: Clock-out capability</a>, <a href="#">Section 8.2.22: Peripheral clocks enable</a>, <a href="#">Section 8.3: Low-power modes</a>, <a href="#">Section 8.4.1: RCC clock control register (RCC_CR)</a>, <a href="#">Section 8.4.9: RCC SMPS step-down converter control register (RCC_SMPSCR)</a>, <a href="#">Section 8.4.16: RCC APB3 peripheral reset register (RCC_APB3RSTR)</a>, <a href="#">Section 8.4.19: RCC AHB3 and AHB4 peripheral clock enable register (RCC_AHB3ENR)</a>, <a href="#">Section 8.4.22: RCC APB2 peripheral clock enable register (RCC_APB2ENR)</a>, <a href="#">Section 8.4.29: RCC peripherals independent clock configuration register (RCC_CCIPR)</a>, <a href="#">Section 8.4.31: RCC control/status register (RCC_CSR)</a>, <a href="#">Section 8.4.34: RCC extended clock recovery register (RCC_EXTCFGR)</a>, <a href="#">Section 8.4.35: RCC CPU2 AHB1 peripheral clock enable register (RCC_C2AHB1ENR)</a>, <a href="#">Section 8.4.37: RCC CPU2 AHB3 and AHB4 peripheral clock enable register (RCC_C2AHB3ENR)</a>, <a href="#">Section 8.4.41: RCC CPU2 APB3 peripheral clock enable register (RCC_C2APB3ENR)</a>, <a href="#">Section 8.4.42: RCC CPU2 AHB1 peripheral clocks enable in Sleep modes register (RCC_C2AHB1SMENR)</a> and <a href="#">Section 8.4.48: RCC CPU2 APB3 peripheral clock enable in Sleep mode register (RCC_C2APB3SMENR)</a>.</p> <p>Updated <a href="#">Section 8.3.1: General-purpose I/O (GPIO)</a>, <a href="#">Section 8.3.2: I/O pin alternate function multiplexer and mapping</a>, <a href="#">Section 8.4.1: GPIO port mode register (GPIOx_MODER)</a> (<math>x = A</math> to <math>C</math> and <math>E</math> and <math>H</math>) and sections from <a href="#">8.4.4</a> to <a href="#">8.4.11</a>.</p> <p>Updated <a href="#">Section 10.1: SYSCFG main features</a>, <a href="#">Section 10.2.7: SYSCFG SRAM2 control and status register (SYSCFG_SCSR)</a>, <a href="#">Section 10.2.16: SYSCFG secure IP control register (SYSCFG_SIPCR)</a> and <a href="#">Table 46: SYSCFG register map and reset values</a>.</p> <p>Updated <a href="#">Table 61: CPU1 vector table</a>, <a href="#">Table 62: CPU2 vector table</a> and <a href="#">Table 63: Wakeup interrupt table</a>.</p> <p>Updated title of <a href="#">Section 14: Extended interrupt and event controller (EXTI)</a>, <a href="#">Section 14.4: EXTI functional description</a>, sections <a href="#">14.6.1</a> to <a href="#">14.6.8</a>, and replaced former sections <a href="#">12.5.9</a> to <a href="#">12.5.12</a> with new sections <a href="#">14.6.9</a> to <a href="#">14.6.16</a>.</p> <p>Updated <a href="#">Table 70: EXTI register map and reset values</a> and <a href="#">Table 63: Wakeup interrupt table</a>.</p> <p>Removed former sections <a href="#">14.5.13</a> to <a href="#">14.5.18</a>.</p> <p>Updated <a href="#">Section 15.1: Introduction</a>, <a href="#">Section 15.2: QUADSPI main features</a>, <a href="#">Section 15.3.2: QUADSPI pins</a>, <a href="#">Section 15.3.8: QUADSPI flash memory configuration</a>, <a href="#">Section 15.3.10: QUADSPI configuration</a>, <a href="#">Section 15.3.15: NCS behavior</a>, <a href="#">Section 15.5.1: QUADSPI control register (QUADSPI_CR)</a> and <a href="#">Section 15.5.6: QUADSPI communication configuration register (QUADSPI_CCR)</a></p> <p>Updated <a href="#">Table 71: QUADSPI pins</a> and <a href="#">Table 73: QUADSPI register map and reset values</a>.</p> <p>Removed former <a href="#">Dual-flash mode</a> and former sections <a href="#">23.7.5</a> to <a href="#">23.7.7</a>.</p>

Table 285. Document revision history (continued)

Date	Revision	Changes
12-Apr-2018	2 (cont'd)	<p>Updated <a href="#">Section 16.7.5: ADC configuration register 2 (ADC_CFGR2)</a>.      Updated <a href="#">Figure 40: ADC block diagram</a> and <a href="#">Figure 48: Stopping ongoing regular conversions</a>.</p> <p>Updated <a href="#">RSA encryption/decryption principle</a> and <a href="#">Table 158: PKA register map and reset values</a>.</p> <p>Updated <a href="#">Section 24.3.29: Debug mode</a>.</p> <p>Updated notes in <a href="#">Using one timer to start another timer</a> and in <a href="#">TIM2 slave mode control register (TIM2_SMCR)</a>.</p> <p>Updated <a href="#">Section 31.3.4: How to program the watchdog timeout</a>, <a href="#">Section 31.5.2: WWDG configuration register (WWDG_CFR)</a> and <a href="#">Table 186: WWDG register map and reset values</a>.</p> <p>Updated <a href="#">Selecting the clock source and the appropriate oversampling method</a>, <a href="#">Section 34.7.4: LPUART control register 3 (LPUART_CR3)</a>, sections <a href="#">34.7.7 to 34.7.12</a> and <a href="#">Table 178: USART interrupt requests</a>.</p> <p>Updated <a href="#">Frame length</a>, <a href="#">Section 36.4.8: SAI clock generator</a> and its subsections, <a href="#">Wrong clock configuration in master mode (with NODIV = 0)</a>, and made two different sections for similar registers in <a href="#">Section 36.6: SAI registers</a>.</p> <p>Updated <a href="#">Table 227: Clock generator programming examples</a> and <a href="#">Table 233: SAI interrupt sources</a>.</p> <p>Updated <a href="#">Table 237: IPCC register map and reset values</a>.</p> <p>Removed former sections <a href="#">37.4.9</a> to <a href="#">38.4.13</a>.</p> <p>Updated <a href="#">Table 240: HSEM register map and reset values</a>.</p> <p>Removed former sections <a href="#">38.4.9</a> to <a href="#">38.4.13</a>.Removed former sections <a href="#">38.4.9</a> to <a href="#">38.4.13</a>.</p> <p>Updated <a href="#">Figure 410: CRS block diagram</a>.</p> <p>Updated <a href="#">Section 41.4.1: JTAG debug port</a>, <a href="#">Section 41.5: Access ports</a>, <a href="#">Section 41.8: Microcontroller debug unit (DBGMCU)</a>, <a href="#">Section 41.8.1: DBGMCU identity code register (DBGMCU_IDCODE)</a> and <a href="#">Section 41.8.2: DBGMCU configuration register (DBGMCU_CR)</a>.</p> <p>Updated <a href="#">Table 271: DBGMCU register map and reset values</a>.</p>
08-Jan-2019	3	<p>Updated <a href="#">Table 1: Memory map and peripheral register boundary addresses</a>.</p> <p>Updated <a href="#">Secure system flash memory programming</a>, <a href="#">Section 3.3.7: Flash main memory erase sequences</a>, <a href="#">Flash memory mass erase</a>, <a href="#">Section 3.3.8: Flash main memory programming sequences</a>, <a href="#">User and read protection option bytes</a>, <a href="#">Secure flash memory start address option bytes</a>, <a href="#">Secure SRAM2 start address and CPU2 reset vector option bytes</a>, <a href="#">Secure user options</a>, <a href="#">WRP Area A address option bytes</a>, <a href="#">WRP Area B address option bytes</a>, <a href="#">Changing the CPU2 security mode</a>, <a href="#">Level 2: No debug</a>, <a href="#">Changing the Read protection level</a>, <a href="#">Section 3.6.4: CPU2 security (ESE)</a>, <a href="#">CPU2 secure SRAM2 areas</a>, <a href="#">CPU2 debug access</a>, <a href="#">Section 3.10.5: Flash memory control register (FLASH_CR)</a>, <a href="#">Section 3.10.7: Flash memory option register (FLASH_OPTR)</a>, <a href="#">Section 3.10.18: Secure flash memory start address register (FLASH_SFR)</a>, <a href="#">Section 3.10.19: Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)</a> and <a href="#">Section 3.10.17: Flash memory CPU2 control register (FLASH_C2CR)</a>.</p> <p>Added <a href="#">Section 3.9: Register access protection</a>.</p>

Table 285. Document revision history (continued)

Date	Revision	Changes
08-Jan-2019	3 (cont'd)	<p>Updated <a href="#">Table 3: Flash memory - Single bank organization</a>, <a href="#">Table 5: Page erase overview</a>, <a href="#">Table 6: Mass erase overview</a>, <a href="#">Table 8: Option bytes format</a>, <a href="#">Table 9: Option bytes organization</a>, <a href="#">Table 13: RDP regression from Level 1 to Level 0 and memory erase</a> and <a href="#">Table 18: Flash interface register map and reset values</a>.</p> <p>Updated titles of sections <a href="#">5.4.1</a> to <a href="#">5.4.5</a>.</p> <p>Updated <a href="#">Section 6.1.1: Independent analog peripherals supply</a>, <a href="#">Section 6.4: Low-power modes</a>, <a href="#">Section 6.6.3: PWR control register 3 (PWR_CR3)</a>, <a href="#">Section 6.6.5: PWR status register 1 (PWR_SR1)</a>, sections <a href="#">6.6.8</a> to <a href="#">6.6.20</a>, <a href="#">6.6.22</a> and <a href="#">6.6.23</a>.</p> <p>Updated <a href="#">Table 23: Sub-system low power wakeup sources</a>, <a href="#">Table 24: Low-power mode summary</a> and <a href="#">Table 25: Functionalities depending on system operating mode</a>.</p> <p>Updated <a href="#">Table 37: Peripherals interconnect matrix</a>.</p> <p>Updated <a href="#">Section 8.2: Clocks</a>, <a href="#">Section 8.2.8: LSI2 clock, LSI2 trimming parameter</a>, <a href="#">Section 8.2.20: Clock-out capability</a>, <a href="#">Section 8.3: Low-power modes</a>, <a href="#">Section 8.4.1: RCC clock control register (RCC_CR)</a> and <a href="#">Section 8.4.31: RCC control/status register (RCC_CSR)</a>.</p> <p>Updated <a href="#">Table 40: Peripheral clock enable</a>, <a href="#">Table 42: RCC register map and reset values</a> and added <a href="#">Table 41: Single core Low power debug configurations</a>.</p> <p>Updated <a href="#">Section 10.2.16: SYSCFG secure IP control register (SYSCFG_SIPCR)</a>.</p> <p>Updated <a href="#">Section 12.4.3: DMAMUX channels</a>, <a href="#">Section 12.4.3: DMAMUX channels, Synchronization overrun and interrupt</a>, <a href="#">Trigger overrun and interrupt</a>, <a href="#">Section 12.6.1: DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)</a>, <a href="#">Section 12.6.3: DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR)</a> and <a href="#">Section 12.6.6: DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR)</a>.</p> <p>Updated <a href="#">Figure 26: DMAMUX block diagram</a>.</p> <p>Replaced AIEC with EXTI in <a href="#">Section 13: Nested vectored interrupt controller (NVIC)</a>, and ID with CoreID in <a href="#">Section 38: Hardware semaphore (HSEM)</a>.</p> <p>Updated <a href="#">Table 61: CPU1 vector table</a> and <a href="#">Table 62: CPU2 vector table</a>.</p> <p>Updated <a href="#">Section 15.2: QUADSPI main features, FIFO and data management</a>, <a href="#">Section 15.3.10: QUADSPI configuration</a>, <a href="#">Section 15.3.11: QUADSPI use</a>, <a href="#">Section 15.5.1: QUADSPI control register (QUADSPI_CR)</a>, <a href="#">Section 15.5.3: QUADSPI status register (QUADSPI_SR)</a> and <a href="#">Section 15.5.6: QUADSPI communication configuration register (QUADSPI_CCR)</a>.</p> <p>Updated <a href="#">Table 73: QUADSPI register map and reset values</a>.</p>

Table 285. Document revision history (continued)

Date	Revision	Changes
08-Jan-2019	3 (cont'd)	<p>Updated <a href="#">Section 16.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)</a>, <a href="#">Section 16.4.7: Single-ended and differential input channels</a>, <a href="#">Section 16.4.8: Calibration (ADCAL, ADCALDIF, ADC_CALFACT)</a>, <a href="#">Section 16.4.10: Constraints when writing the ADC control bits</a>, <a href="#">Section 16.4.11: Channel selection (SQRx, JSQRx)</a> and <a href="#">Section 16.4.12: Channel-wise programmable sampling time (SMPR1, SMPR2)</a>.</p> <p>Updated <a href="#">Table 75: ADC internal input/output signals</a> and <a href="#">Figure 42: ADC1 connectivity</a>.</p> <p>Updated <a href="#">Section 21.1: Introduction</a>, <a href="#">Section 21.2: RNG main features</a> and <a href="#">Health checks</a>.</p> <p>Updated <a href="#">Montgomery space and fast mode operations</a>, <a href="#">RSA encryption/decryption principle</a>, <a href="#">Executing a PKA operation</a>, <a href="#">Using precomputed Montgomery parameters (PKA fast mode)</a>, <a href="#">Section 23.3.7: PKA error management</a>, <a href="#">Extended ECDSA support</a>, <a href="#">Section 23.5.1: Supported elliptic curves</a>, <a href="#">Section 23.7.1: PKA control register (PKA_CR)</a>, <a href="#">Section 23.7.2: PKA status register (PKA_SR)</a>, <a href="#">Section 23.7.3: PKA clear flag register (PKA_CLRFR)</a> and <a href="#">Section 23.7.4: PKA RAM</a>.</p> <p>Updated <a href="#">Table 130: Modular addition</a>, <a href="#">Table 131: Modular subtraction</a>, <a href="#">Table 132: Montgomery multiplication</a>, <a href="#">Table 136: Modular reduction</a>, <a href="#">Table 137: Arithmetic addition</a>, <a href="#">Table 138: Arithmetic subtraction</a>, <a href="#">Table 139: Arithmetic multiplication</a>, <a href="#">Table 140: Arithmetic comparison</a>, <a href="#">Table 175: ECC curves parameters</a>, <a href="#">Table 151: Modular exponentiation computation times</a>, <a href="#">Table 152: ECC scalar multiplication computation times</a>, <a href="#">Table 153: ECDSA signature average computation times</a>, <a href="#">Table 154: ECDSA verification average computation times</a> and <a href="#">Table 158: PKA register map and reset values</a>.</p> <p>Replaced BKE<sub>x</sub>, BKPx<sub>y</sub> by BKE, BK2E, BKP, BK2P and split registers CCMR1 and CCMR2 in <a href="#">Section 24: Advanced-control timer (TIM1)</a>.</p> <p>Updated <a href="#">Section 24.3.16: Using the break function</a>.</p> <p>Updated <a href="#">Figure 144: Advanced-control timer block diagram</a> and <a href="#">Figure 191: Output redirection (BRK2 request not represented)</a>.</p> <p>Updated <a href="#">Section 26.3.19: Debug mode</a> and <a href="#">Section 26.4.6: TIM<sub>x</sub> capture/compare mode register 1 [alternate] (TIM<sub>x</sub>_CCMR1) (x = 16 to 17)</a>.</p> <p>Updated <a href="#">Table 171: TIM16/TIM17 register map and reset values</a>.</p> <p>Updated <a href="#">Figure 275: Output redirection</a>.</p> <p>Updated <a href="#">Section 29.2: RTC main features</a> and <a href="#">Section 29.7.20: RTC backup registers (RTC_BKPxR)</a>.</p> <p>Updated <a href="#">Table 184: RTC register map and reset values</a>.</p> <p>Added <a href="#">Section 30.3.4: Low-power freeze</a>.</p> <p>Updated <a href="#">Section 49.3.1: IWDG block diagram</a> and <a href="#">Section 30.4.2: IWDG prescaler register (IWDG_PR)</a>.</p> <p>Removed former <a href="#">Section 49.3.6: Behavior in Stop and Standby modes</a>.</p> <p>Updated <a href="#">Table 185: IWDG register map and reset values</a>.</p> <p>Updated <a href="#">Section 38.3.8: AHB bus master ID verification</a> and Sections <a href="#">38.4.1</a> to <a href="#">32.4.7</a>.</p> <p>Added <a href="#">Table 239: Authorized AHB bus master IDs</a> and updated <a href="#">Table 240: HSEM register map and reset values</a>.</p>

Table 285. Document revision history (continued)

Date	Revision	Changes
08-Jan-2019	3 (cont'd)	<p>Updated <a href="#">Section 41.1: Introduction</a>, <a href="#">Section 41.4.8: DP target identification register (DP_TARGETIDR)</a>, <a href="#">Section 41.7: Cross trigger interface registers</a>, <a href="#">Section 41.8.1: DBGMCU identity code register (DBGMCU_IDCODE)</a>, <a href="#">Section 41.10.14: DWT CoreSight peripheral identity register 1 (DWT_PIDR1)</a> and <a href="#">Section 41.14.14: DWT CoreSight peripheral identity register 1 (DWT_PIDR1)</a>.</p> <p>Updated <a href="#">Table 264: Debug port register map and reset values</a> and <a href="#">Table 271: DBGMCU register map and reset values</a>.</p> <p>Removed former <a href="#">Section 32.11: CPU2 Instrumentation Trace Macrocell (ITM)</a> and its subsections.</p> <p>Updated <a href="#">Section 42.1: Unique device ID register (96 bits)</a></p>
01-Mar-2019	4	<p>Changed document classification, from ST restricted to Public.</p> <p>Updated <a href="#">Section 8.3: Low-power modes</a>, <a href="#">Channel configuration procedure</a>, <a href="#">Channel state and disabling a channel</a>, <a href="#">Section 21.1: Introduction</a>, <a href="#">Section 21.2: RNG main features</a>, <a href="#">Section 21.3.3: Random number generation</a> and its subsections, <a href="#">Section 21.3.4: RNG initialization</a>, <a href="#">Section 21.3.5: RNG operation</a> and its subsections, <a href="#">Section 21.3.6: RNG clocking</a>, <a href="#">Section 21.3.7: Error management</a>, <a href="#">Section 21.3.8: RNG low-power usage</a>, <a href="#">Section 21.6.1: Introduction</a>, <a href="#">Section 21.7: RNG registers</a>, <a href="#">Section 21.7.2: RNG status register (RNG_SR)</a>, <a href="#">Section 21.7.3: RNG data register (RNG_DR)</a>, sections <a href="#">25.4.7 to 25.4.24</a>, <a href="#">26.4.6 and 26.4.7</a>, <a href="#">Section 33.5.4: USART FIFOs and thresholds</a>, <a href="#">Section 34.4.4: LPUART FIFOs and thresholds</a> and <a href="#">Section 50.4.14: SPDIF output</a>.</p> <p>Updated <a href="#">Table 63: Wakeup interrupt table</a>, <a href="#">Table 168: TIM2 register map and reset values</a> and <a href="#">Table 260: STM32WB55xx SPI implementation</a>.</p> <p>Updated <a href="#">Figure 113: RNG block diagram</a>, <a href="#">Figure 114: Entropy source model</a> and <a href="#">Figure 355: Reception using DMA</a>.</p> <p>Added <a href="#">Figure 115: RNG initialization overview</a> and footnote to <a href="#">Figure 370: Packing data in FIFO for transmission and reception</a>.</p>
13-Feb-2020	5	<p>Updated <a href="#">Section 2.1: System architecture</a>, <a href="#">Section 3.5: FLASH UID64</a>, <a href="#">Section 3.6.4: CPU2 security (ESE)</a>, <a href="#">Section 6.4.12: Auto wakeup from Low-power mode</a>, <a href="#">Section 6.6.8: PWR control register 5 (PWR_CR5)</a>, <a href="#">Section 7.4.7: From USB to timer (TIM2)</a>, <a href="#">Software reset</a>, <a href="#">Section 8.2: Clocks</a>, <a href="#">External source</a>, <a href="#">Section 8.2.2: HSI16 clock</a>, <a href="#">Section 8.2.3: MSI clock</a>, <a href="#">Section 8.2.5: PLLs</a>, <a href="#">Section 8.2.8: LSI2 clock</a>, <a href="#">Section 8.4.1: RCC clock control register (RCC_CR)</a>, <a href="#">Section 8.4.4: RCC PLL configuration register (RCC_PLLCFGR)</a>, <a href="#">Section 8.4.31: RCC control/status register (RCC_CSR)</a>, <a href="#">Section 8.3: GPIO functional description</a>, <a href="#">Section 8.3.2: I/O pin alternate function multiplexer and mapping</a>, <a href="#">Section 8.3.7: I/O alternate function input/output</a>, <a href="#">Section 10.2.15: SYSCFG CPU2 interrupt mask register 2 (SYSCFG_C2IMR2)</a>, <a href="#">Section 13.1: NVIC main features</a>, <a href="#">Section 16.4.32: Monitoring the internal voltage reference</a>, <a href="#">Section 29.5: RTC low-power modes</a>, <a href="#">Section 36.4.10: PDM interface</a>, <a href="#">Section 40.7.2: CRS configuration register (CRS_CFGR)</a> and <a href="#">Section 42.3: Package data register</a>.</p> <p>Added note in <a href="#">Section 3.8: FLASH interrupts</a> and <a href="#">Section 40.3: CRS implementation</a>.</p> <p>Removed note from <a href="#">Section 19.2: LCD main features</a>.</p>

Table 285. Document revision history (continued)

Date	Revision	Changes
13-Feb-2020	5 (cont'd)	<p>Updated <a href="#">Table 1: Memory map and peripheral register boundary addresses</a>, <a href="#">Table 2: Boot modes</a>, <a href="#">Table 25: Functionalities depending on system operating mode</a>, <a href="#">Table 42: RCC register map and reset values</a>, <a href="#">Table 61: CPU1 vector table</a>, <a href="#">Table 62: CPU2 vector table</a> and <a href="#">Table 63: Wakeup interrupt table</a>.</p> <p>Updated <a href="#">Figure 15: Clock tree</a>, <a href="#">Figure 19: Three-volt or Five-volt tolerant GPIO structure (TT or FT)</a>, <a href="#">Figure 20: Input floating / pull up / pull down configurations</a>, <a href="#">Figure 21: Output configuration</a> and <a href="#">Figure 22: Alternate function configuration</a>, and removed former <a href="#">Figure 21: Basic structure of a 5-Volt tolerant I/O port bit</a>.</p>
11-Jul-2020	6	<p>Introduced STM32WB35xx devices.</p> <p>Updated <a href="#">Introduction</a>, <a href="#">Section 3.3.1: Flash memory organization</a>, <a href="#">PCROP1A start address option bytes</a>, <a href="#">PCROP1A end address option bytes</a>, <a href="#">WRP Area A address option bytes</a>, <a href="#">WRP Area B address option bytes</a>, <a href="#">PCROP1B start address option bytes</a>, <a href="#">PCROP1B end address option bytes</a>, <a href="#">Section 3.5: FLASH UID64</a>, <a href="#">Section 3.10.5: Flash memory control register (FLASH_CR)</a>, sections <a href="#">3.10.8 to 3.10.13</a>, <a href="#">Section 3.10.17: Flash memory CPU2 control register (FLASH_C2CR)</a>, sections <a href="#">6.6.2 to 6.6.4</a>, <a href="#">Section 6.6.7: PWR status clear register (PWR_SCR)</a>, sections <a href="#">6.6.12 to 6.6.14</a> and <a href="#">6.6.18 to 6.6.20</a>, <a href="#">Section 6.6.22: PWR CPU2 control register 3 (PWR_C2CR3)</a>, <a href="#">Section 7.4.8: From internal analog to ADC1</a>, sections <a href="#">8.4.12 to 8.4.13</a>, <a href="#">8.4.17 to 8.4.18</a>, <a href="#">Section 8.4.20: RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1)</a>, sections <a href="#">8.4.23 to 8.4.24</a>, <a href="#">Section 8.4.26: RCC APB1 peripheral clocks enable in Sleep mode register 1 (RCC_APB1SMENR1)</a>, <a href="#">Section 8.4.45: RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 1 (RCC_C2APB1SMENR1)</a>, <a href="#">Section 8.4.48: RCC CPU2 APB3 peripheral clock enable in Sleep mode register (RCC_C2APB3SMENR)</a>, <a href="#">Section 9.4.3: I/O port control registers</a>, sections <a href="#">10.2.3 to 10.2.7</a>, <a href="#">Section 10.2.15: SYSCFG CPU2 interrupt mask register 2 (SYSCFG_C2IMR2)</a>, <a href="#">Section 12.4.4: DMAMUX request line multiplexer</a>, <a href="#">Calculating the actual <math>V_{REF+}</math> voltage using the internal reference voltage</a>, <a href="#">Section 17.5.2: VREFBUF calibration control register (VREFBUF_CCR)</a>. <a href="#">Section 18.6.1: Comparator 1 control and status register (COMP1_CSR)</a>, <a href="#">Section 20.2: TSC main features</a>, <a href="#">Section 20.3.4: Charge transfer acquisition sequence</a>, <a href="#">Montgomery space and fast mode operations</a>, <a href="#">Section 23.7.2: PKA status register (PKA_SR)</a>, <a href="#">Section 16.4.7: Single-ended and differential input channels</a>, <a href="#">Software procedure to enable the ADC</a>, <a href="#">Section 24.4.1: TIM1 control register 1 (TIM1_CR1)</a>, <a href="#">Section 24.4.8: TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)</a>, <a href="#">Section 24.4.20: TIM1 break and dead-time register (TIM1_BDTR)</a>, sections <a href="#">24.4.27 to 24.4.28</a>, <a href="#">Section 25.3.19: Timer synchronization</a>, <a href="#">Section 25.4.8: TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1)</a>, sections <a href="#">27.7.1 to 27.7.2</a>, <a href="#">Section 34.7.8: LPUART interrupt and status register [alternate] (LPUART_ISR)</a>, <a href="#">Section 36.6.17: SAI PDM control register (SAI_PDMCR)</a>, <a href="#">Section 41.1: Introduction</a>, <a href="#">Section 41.8.2: DBGMCU configuration register (DBGMCU_CR)</a>, <a href="#">Section 41.8.4: DBGMCU CPU2 APB1 peripheral freeze register 1 (DBGMCU_C2APB1FZR1)</a>, <a href="#">Section 41.8.6: DBGMCU CPU2 APB1 peripheral freeze register 2 (DBGMCU_C2APB1FZR2)</a> and <a href="#">Section 41.8.8: DBGMCU CPU2 APB2 peripheral freeze register (DBGMCU_C2APB2FZR)</a>.</p>

Table 285. Document revision history (continued)

Date	Revision	Changes
11-Jul-2020	6 (cont'd)	<p>Updated <a href="#">Table 3: Flash memory - Single bank organization</a>, <a href="#">Table 95: COMP1 input plus assignment</a>, <a href="#">Table 96: COMP1 input minus assignment</a>, <a href="#">Table 151: Modular exponentiation computation times</a>, <a href="#">Table 156: Montgomery parameters average computation times</a>, <a href="#">Table 163: Output control bits for complementary OCx and OCxN channels with break feature</a>, <a href="#">Table 167: Output control bit for standard OCx channels</a> and <a href="#">Table 170: Output control bits for complementary OCx and OCxN channels with break feature</a> (TIM16/17).</p> <p>Updated <a href="#">Figure 167: Control circuit in normal mode, internal clock divided by 1</a>, <a href="#">Figure 173: Capture/compare channel 1 main circuit</a>, <a href="#">Figure 204: General-purpose timer block diagram</a>, <a href="#">Figure 230: Capture/Compare channel 1 main circuit</a>, <a href="#">Figure 253: TIM16/TIM17 block diagram</a> and <a href="#">Figure 267: Capture/compare channel 1 main circuit</a>.</p> <p>Added <a href="#">Section 7.2: Interconnect matrix implementation</a>, <a href="#">Note: to Section 8.4.3: RCC clock configuration register (RCC_CFGR)</a>, <a href="#">Section 9.3: GPIO implementation</a>, <a href="#">Section 13.2: NVIC implementation</a>, <a href="#">Section 14.2: EXTI implementation</a>, <a href="#">Section 16.3: ADC implementation</a>, <a href="#">Section 17.2: VREFBUF implementation</a>, <a href="#">Section 26.3.18: Using timer output as trigger for other timers (TIM16/TIM17)</a>, <a href="#">Section 29.3: RTC implementation</a> and <a href="#">Section 36.3: SAI implementation</a>.</p> <p>Added <a href="#">Table 52: DMAMUX implementation</a>, <a href="#">Table 155: Point on elliptic curve Fp check average computation times</a>, <a href="#">Table 179: LPTIM implementation</a>, <a href="#">Table 187: I2C implementation</a> and footnote to <a href="#">Table 220: SPI implementation</a>.</p> <p>Minor text edits across the whole document.</p>
20-Apr-2021	7	<p>Updated <a href="#">Figure 109: Surface charge transfer analog I/O group structure</a>, <a href="#">Figure 343: RS232 RTS flow control</a>, <a href="#">Figure 344: RS232 CTS flow control</a>, <a href="#">Figure 385: Detailed PDM interface block diagram</a> and <a href="#">Figure 400: IPCC Simplex - Send procedure state diagram</a>.</p> <p>Updated <a href="#">Table 107: Acquisition sequence summary</a>, <a href="#">Table 150: Family of supported curves for ECC operations</a>, <a href="#">Table 157: PKA interrupt requests</a>, <a href="#">Table 213: Error calculation for programmed baud rates at <math>f_{uart\_ker\_ck\_pres} = 32.768 \text{ kHz}</math></a>, <a href="#">Table 214: Error calculation for programmed baud rates at <math>f_{CK} = 100 \text{ MHz}</math></a> and <a href="#">Table 238: HSEM internal input/output signals</a></p> <p>Added <a href="#">Table 235: IPCC interface signals</a>.</p> <p>Added footnotes to <a href="#">Table 132: Montgomery multiplication</a>, <a href="#">Table 223: STM32WB55xx SAI features</a>, <a href="#">Table 225: SAI input/output pins</a> and to <a href="#">Figure 377: SAI functional block diagram</a>, <a href="#">Figure 385: Detailed PDM interface block diagram</a>.</p> <p>Added <a href="#">Section 33.6: USART in low-power modes</a> and <a href="#">Section 34.5: LPUART in low-power modes</a>.</p> <p>Added notes in <a href="#">Section 27.4.6: Trigger multiplexer</a> and <a href="#">Section 20.3.2: Surface charge transfer acquisition overview</a>.</p>

Table 285. Document revision history (continued)

Date	Revision	Changes
20-Apr-2021	7 (cont'd)	<p>Updated <a href="#">Introduction</a>, <a href="#">Related documents</a>, ST production values in <a href="#">Section 3.4.1: Option bytes description</a>, <a href="#">Section 3.10.6: Flash memory ECC register (FLASH_ECCR)</a>, <a href="#">Section 3.10.9: Flash memory PCROP zone A end address register (FLASH_PCROP1AER)</a>, <a href="#">Section 3.10.10: Flash memory WRP area A address register (FLASH_WRP1AR)</a>, <a href="#">Section 3.10.11: Flash memory WRP area B address register (FLASH_WRP1BR)</a>, <a href="#">Section 3.10.12: Flash memory PCROP zone B start address register (FLASH_PCROP1BSR)</a>, <a href="#">Section 3.10.13: Flash memory PCROP zone B end address register (FLASH_PCROP1BER)</a>, <a href="#">Section 3.10.14: Flash memory IPCC mailbox data buffer address register (FLASH_IPCCBR)</a>, <a href="#">Section 3.10.18: Secure flash memory start address register (FLASH_SFR)</a>, <a href="#">Section 3.10.19: Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)</a>, <a href="#">Section 4.1: Introduction</a>, <a href="#">Section 4.2: Main features</a>, <a href="#">Section 5.2: CRC main features</a>, <a href="#">Polynomial programmability</a>, <a href="#">Section 5.4: CRC registers</a>, <a href="#">Section 8.2: Clocks</a>, <a href="#">Section 8.2.14: SMPS step-down converter clock</a>, <a href="#">Section 8.4.1: RCC clock control register (RCC_CR)</a>, <a href="#">Section 8.4.3: RCC clock configuration register (RCC_CFGR)</a>, <a href="#">Section 16.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)</a>, note in <a href="#">Triggered injection mode</a>, <a href="#">Section 20.2: TSC main features</a>, <a href="#">Section 23.3.4: PKA public key acceleration</a>, <a href="#">Section 23.5.1: Supported elliptic curves</a>, <a href="#">Section 36.4.10: PDM interface</a>, <a href="#">Section 27.7.1: LPTIM interrupt and status register (LPTIM_ISR)</a>, <a href="#">Section 33.2: USART main features</a>, <a href="#">Section 33.8.9: USART interrupt and status register (USART_ISR)</a>, <a href="#">Section 33.8.10: USART interrupt and status register [alternate] (USART_ISR)</a>, <a href="#">Section 34.4.14: LPUART low-power management</a>, <a href="#">Section 36.4.10: PDM interface</a>, <a href="#">Section 36.6.18: SAI PDM delay register (SAI_PDMDDLY)</a>, <a href="#">Section 38.3.3: HSEM lock procedures</a>, <a href="#">Section 38.3.5: HSEM unlock procedures</a>, <a href="#">Section 38.3.6: HSEM COREID semaphore clear</a>, <a href="#">Section 38.3.7: HSEM interrupts</a>, <a href="#">Section 38.3.8: AHB bus master ID verification</a>, <a href="#">Section 41.4.7: DP data link control register (DP_DLCSR)</a>, <a href="#">Simplex communications</a>, <a href="#">Section 41.5.1: AP control/status word register (AP_CSWR)</a>, <a href="#">Section 41.5.5: AP base address register (AP_BASER)</a>, <a href="#">Section 41.5.6: AP identification register (AP_IDR)</a> and <a href="#">Section 41.8.2: DBGMCU configuration register (DBGMCU_CR)</a>.</p> <p>Minor text edits across the whole document.</p>

Table 285. Document revision history (continued)

Date	Revision	Changes
13-Aug-2021	8	<p>Updated <a href="#">Section 3.5: FLASH UID64</a>, <a href="#">Section 5.2: CRC main features</a>, <a href="#">Section 6.6.2: PWR control register 2 (PWR_CR2)</a>, <a href="#">Section 8.2.8: LS12 clock</a>, <a href="#">Section 8.2.12: Clock security system on LSE (LSECSS)</a>, <a href="#">Section 9.4.2: I/O pin alternate function multiplexer and mapping</a>, <a href="#">Section 24.3.16: Using the break function</a>, <a href="#">Section 24.4.8: TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)</a>, <a href="#">Section 25.4.8: TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1)</a>, <a href="#">Section 26.3.11: Using the break function</a>, <a href="#">Section 29.4.14: Calibration clock output</a>, <a href="#">Section 33.8.3: USART control register 2 (USART_CR2)</a> and <a href="#">Section 41.8.1: DBGMCU identity code register (DBGMCU_IDCODE)</a>.</p> <p>Added <a href="#">Section 26.3.13: 6-step PWM generation</a>.</p> <p>Changed SCLK into CK throughout <a href="#">Section 33: Universal synchronous/asynchronous receiver transmitter (USART/UART)</a>.</p> <p>Minor text edits across the whole document.</p> <p>Added footnote 2 to <a href="#">Table 21: Supply configuration control</a>.</p> <p>Updated <a href="#">Figure 229: TDM frame configuration examples</a> and <a href="#">Table 271: DBGMCU register map and reset values</a>.</p> <p>Updated <a href="#">Figure 144: Advanced-control timer block diagram</a>, <a href="#">Figure 167: Control circuit in normal mode, internal clock divided by 1</a>, <a href="#">Figure 263: Control circuit in normal mode, internal clock divided by 1</a>, <a href="#">Figure 323: TC/TXE behavior when transmitting</a>, <a href="#">Figure 324: Start bit detection when oversampling by 16 or 8</a>, <a href="#">Figure 332: USART example of synchronous master transmission</a> and footnotes of <a href="#">Figure 385: Detailed PDM interface block diagram</a>.</p>
04-Oct-2021	9	<p>Updated <a href="#">Section 3.5: FLASH UID64</a>, <a href="#">DMA operation in different operating modes</a>, <a href="#">Section 33.5.20: RS232 Hardware flow control and RS485 Driver Enable</a>, <a href="#">Section 33.8.4: USART control register 3 (USART_CR3)</a>, <a href="#">Section 34.4.13: RS232 Hardware flow control and RS485 Driver Enable</a>, <a href="#">Section 34.7.4: LPUART control register 3 (LPUART_CR3)</a> and <a href="#">Section 41.8.1: DBGMCU identity code register (DBGMCU_IDCODE)</a>.</p> <p>Updated <a href="#">Figure 135: GCM authenticated encryption</a>.</p> <p>Added <a href="#">Section 42.4: Part number codification register</a>.</p> <p>Minor text edits across the whole document.</p>

Table 285. Document revision history (continued)

Date	Revision	Changes
07-Jun-2022	10	<p>Updated <a href="#">Introduction</a>, <a href="#">Section 3.3.1: Flash memory organization</a>, <a href="#">Section 3.4.1: Option bytes description</a>, <a href="#">Section 4.1: Introduction</a>, <a href="#">Section 4.2: Main features</a>, <a href="#">Section 6.4.4: Exiting Low-power mode</a>, <a href="#">Section 8.4.5: RCC PLLSAI1 configuration register (RCC_PLLSAI1CFGR)</a>, <a href="#">Section 8.4.30: RCC backup domain control register (RCC_BDCR)</a>, <a href="#">Section 8.4.31: RCC control/status register (RCC_CSR)</a>, <a href="#">Section 8.4.33: RCC clock HSE register (RCC_HSECR)</a>, <a href="#">Analog watchdog</a>, <a href="#">Triggering the start of a command</a>, <a href="#">Section 16.4.6: ADC Deep-power-down mode (DEEPPWD)</a> and <a href="#">ADC voltage regulator (ADVREGEN)</a>, <a href="#">Reading the temperature</a>, <a href="#">Section 21.2: RNG main features</a>, <a href="#">Section 41.7.3: CTI application trigger set register (CTI_APPSETR)</a>, <a href="#">Section 41.7.15: CTI lock access register (CTI_LAR)</a>, and <a href="#">Section 41.8.2: DBGMCU configuration register (DBGMCU_CR)</a>.</p> <p>Added <a href="#">Section 16.5: ADC in low-power mode</a> and <a href="#">Section 43: Important security notice</a>.</p> <p>Updated <a href="#">Figure 57: Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs during an ongoing conversion</a>, <a href="#">Figure 59: Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs outside an ongoing conversion</a>, <a href="#">Figure 87: Triggered regular oversampling with injection</a>, <a href="#">Figure 135: GCM authenticated encryption</a>, <a href="#">Figure 299: Transfer bus diagrams for I2C slave transmitter (mandatory events only)</a>, <a href="#">Figure 302: Transfer bus diagrams for I2C slave receiver (mandatory events only)</a>, and <a href="#">Figure 309: Transfer bus diagrams for I2C master transmitter (mandatory events only)</a>.</p> <p>Updated <a href="#">Table 65: ADC input/output pins</a> and <a href="#">Table 222: SPI register map and reset values</a>.</p> <p>Minor text edits across the whole document.</p>

# Index

## A

ADC_AWD2CR	499
ADC_AWD3CR	500
ADC_CALFACT	501
ADC_CCR	502
ADC_CFGR	483
ADC_CFGR2	487
ADC_CR	480
ADC_CSR	501
ADC_DIFSEL	500
ADC_DR	495
ADC_IER	478
ADC_ISR	476
ADC_JDRy	499
ADC_JSQR	496
ADC_OFRy	498
ADC_SMPR1	488
ADC_SMPR2	489
ADC_SQR1	492
ADC_SQR2	493
ADC_SQR3	494
ADC_SQR4	495
ADC_TR1	490
ADC_TR2	490
ADC_TR3	491
AES_CR	623
AES_DINR	627
AES_DOUTR	627
AES_IVR0	630
AES_IVR1	630
AES_IVR2	630
AES_IVR3	631
AES_KEYR0	628
AES_KEYR1	629
AES_KEYR2	629
AES_KEYR3	629
AES_KEYR4	631
AES_KEYR5	631
AES_KEYR6	632
AES_KEYR7	632
AES_SR	626
AES_SUSPxR	632
AP_BD0-3R	1370
AP_DRWR	1370
AP_TAR	1370

## B

BPU_CIDR0	1434
BPU_CIDR1	1435
BPU_CIDR2	1435
BPU_CIDR3	1436
BPU_COMPxR	1432
BPU_CTRLR	1431
BPU_PIDR0	1433
BPU_PIDR1	1433
BPU_PIDR2	1433
BPU_PIDR3	1434
BPU_PIDR4	1432
BPU_REMAPR	1431

## C

C1ROM_CIDR0	1442
C1ROM_CIDR1	1442
C1ROM_CIDR2	1443
C1ROM_CIDR3	1443, 1445
C1ROM_MEMTYPER	1439
C1ROM_PIDR0	1440
C1ROM_PIDR1	1441
C1ROM_PIDR2	1441
C1ROM_PIDR3	1442
C1ROM_PIDR4	1440
C2ROM1_CIDR0	1407
C2ROM1_CIDR1	1408
C2ROM1_CIDR2	1408
C2ROM1_CIDR3	1408
C2ROM1_MEMTYPER	1405
C2ROM1_PIDR0	1405
C2ROM1_PIDR1	1406
C2ROM1_PIDR2	1406
C2ROM1_PIDR3	1407
C2ROM1_PIDR4	1405
C2ROM2_CIDR0	1413
C2ROM2_CIDR1	1414
C2ROM2_CIDR2	1414
C2ROM2_CIDR3	1414
C2ROM2_MEMTYPER	1411
C2ROM2_PIDR0	1411
C2ROM2_PIDR1	1412
C2ROM2_PIDR2	1412
C2ROM2_PIDR3	1413
C2ROM2_PIDR4	1411
COMP1_CSR	518
COMP2_CSR	520

CRC_CR .....	136	DMA_ISR .....	335
CRC_DR .....	135	DMAMUX_CFR .....	357
CRC_IDR .....	135	DMAMUX_CSR .....	357
CRC_INIT .....	137	DMAMUX_CxCR .....	356
CRC_POL .....	137	DMAMUX_RGCFR .....	359
CRS_CFGR .....	1344	DMAMUX_RGSR .....	359
CRS_CR .....	1343	DMAMUX_RGxCR .....	358
CRS_ICR .....	1347	DP_ABORTR .....	1358
CRS_ISR .....	1345	DP_CTRL/STATR .....	1359
CTI_APPCLEAR .....	1379	DP_DPIDR .....	1358
CTI_APPPULSER .....	1380	DWT_CIDR0 .....	1426, 1455
CTI_APPSETR .....	1378	DWT_CIDR1 .....	1426, 1455
CTI_AUTHSTATR .....	1385	DWT_CIDR2 .....	1426, 1455
CTI_CHINSTSR .....	1382	DWT_CIDR3 .....	1427, 1456
CTI_CHOUTTSR .....	1382	DWT_COMPxR .....	1422, 1451
CTI_CIDR0 .....	1389	DWT_CPICNTR .....	1419, 1448
CTI_CIDR1 .....	1389	DWT_CTRLR .....	1417, 1446
CTI_CIDR2 .....	1390	DWT_CYCCNTR .....	1419, 1448
CTI_CIDR3 .....	1390	DWT_EXCCNTR .....	1420, 1449
CTI_CLAIMCLR .....	1384	DWT_FOLDCNTR .....	1421, 1450
CTI_CLAIMSETR .....	1383	DWT_FUNCTxR .....	1422, 1451
CTI_CONTROLR .....	1378	DWT_LSUCNTR .....	1421, 1450
CTI_DEVIDR .....	1386	DWT_MASKxR .....	1422, 1451
CTI_DEVTYPER .....	1386	DWT_PCSR .....	1421, 1450
CTI_GATER .....	1383	DWT_PIDR0 .....	1424, 1453
CTI_INENRx .....	1380	DWT_PIDR1 .....	1424, 1453
CTI_INTACKR .....	1378	DWT_PIDR2 .....	1425, 1454
CTI_LAR .....	1384	DWT_PIDR3 .....	1425, 1454
CTI_LSR .....	1385	DWT_PIDR4 .....	1423, 1452
CTI_OUTENRx .....	1381	DWT_SLPCNTR .....	1420, 1449
CTI_PIDR0 .....	1387		
CTI_PIDR1 .....	1387		
CTI_PIDR3 .....	1388		
CTI_PIDR4 .....	1387		
CTI_TRGISTSR .....	1381		
CTI_TRGOSTSR .....	1382		
<b>D</b>			
DBGMCU_APB1FZR1 .....	1395	ETM_AUTHSTATR .....	1487
DBGMCU_APB1FZR2 .....	1397	ETM_CCER .....	1481
DBGMCU_APB2FZR .....	1398	ETM_CCR .....	1475
DBGMCU_C2APB1FZR1 .....	1396	ETM_CIDR0 .....	1490
DBGMCU_C2APB1FZR2 .....	1398	ETM_CIDR1 .....	1490
DBGMCU_C2APB2FZR .....	1399	ETM_CIDR2 .....	1491
DBGMCU_CR .....	1394	ETM_CIDR3 .....	1491
DBGMCU_IDCODE .....	1394	ETM CLAIMCLR .....	1485
DMA_CCRx .....	339	ETM CLAIMSETR .....	1485
DMA_CMARx .....	343	ETM_CNTRLDVR1 .....	1480
DMA_CNDTRx .....	342	ETM_CR .....	1474
DMA_CPARx .....	342	ETM_DEVTYPER .....	1487
DMA_IFCR .....	338	ETM_FFLR .....	1479
		ETM_IDR .....	1481
		ETM_IDR2 .....	1484
		ETM_LAR .....	1486
		ETM_LSR .....	1486
		ETM_PDSR .....	1484
		ETM_PIDR0 .....	1488

ETM_PIDR1	1488	FPB_CIDR1	1471
ETM_PIDR2	1489	FPB_CIDR2	1471
ETM_PIDR3	1489	FPB_CIDR3	1472
ETM_PIDR4	1488	FPB_COMPxR	1468
ETM_SCR	1477	FPB_CTRLR	1467
ETM_SR	1477	FPB_PIDR0	1469
ETM_SYNCFR	1480	FPB_PIDR1	1469
ETM_TECR1	1479	FPB_PIDR2	1470
ETM_TEEVR	1478	FPB_PIDR3	1470
ETM_TESSEICR	1482	FPB_PIDR4	1468
ETM_TRACEIDR	1484	FPB_REMAPR	1467
ETM_TRIGGER	1476		
ETM_TSEVR	1483		
EXTI_C2EMR1	385	G	
EXTI_C2EMR2	387	GPIOx_AFRH	304
EXTI_C2IMR1	384	GPIOx_AFRL	303
EXTI_C2IMR2	386	GPIOx_BRR	305
EXTI_EMR1	384	GPIOx_BSRR	301
EXTI_EMR2	386	GPIOx_IDR	300
EXTI_FTSR1	379	GPIOx_LCKR	302
EXTI_FTSR2	381	GPIOx_MODER	298
EXTI_IMR1	383	GPIOx_ODR	301
EXTI_IMR2	385	GPIOx_OSPEEDR	299
EXTI_PR1	380	GPIOx_OTYPER	299
EXTI_PR2	383	GPIOx_PUPDR	300
EXTI_RTSR1	378		
EXTI_RTSR2	380		
EXTI_SWIER1	379	H	
EXTI_SWIER2	382	HSEM_CnICR	1300
		HSEM_CnIER	1300
		HSEM_CnISR	1300
		HSEM_CnMISR	1301
		HSEM_CR	1301
		HSEM_KEYR	1302
		HSEM_RLRx	1299
		HSEM_Rx	1298
		I	
		I2C_CR1	1020
		I2C_CR2	1023
		I2C_ICR	1031
		I2C_ISR	1029
		I2C_OAR1	1025
		I2C_OAR2	1026
		I2C_PECR	1032
		I2C_RXDR	1033
		I2C_TIMEOUTR	1028
		I2C_TIMINGR	1027
		I2C_TXDR	1033
		IPCC_C1CR	1284
		IPCC_C1MR	1285
		IPCC_C1SCR	1286

IPCC_C1TOC2SR	1286
IPCC_C2CR	1287
IPCC_C2MR	1287
IPCC_C2SCR	1288
IPCC_C2TOC1SR	1289
ITM_CIDR0	1464
ITM_CIDR1	1464
ITM_CIDR2	1464
ITM_CIDR3	1465
ITM_PIDR0	1462
ITM_PIDR1	1462
ITM_PIDR2	1463
ITM_PIDR3	1463
ITM_PIDR4	1461
ITM_STIMRx	1459
ITM_TCR	1460
ITM_TER	1459
ITM_TPR	1460
IWDG_KR	956
IWDG_PR	957
IWDG_RLR	958
IWDG_SR	959
IWDG_WINR	960

**L**

LCD_CLR	552
LCD_CR	547
LCD_FCR	548
LCD_RAM	552
LCD_SR	551
LPTIM_ARR	906
LPTIM_CFGR	901
LPTIM_CMP	906
LPTIM_CNT	907
LPTIM_CR	904
LPTIM_ICR	900
LPTIM_IER	900
LPTIM_ISR	899
LPTIM1_OR	907
LPTIM2_OR	908
LPUART_BRR	1162
LPUART_CR1	1151, 1154
LPUART_CR2	1157
LPUART_CR3	1159
LPUART_ICR	1171
LPUART_ISR	1163, 1168
LPUART_PRESC	1173
LPUART_RDR	1172
LPUART_RQR	1163
LPUART_TDR	1172

**P**

PKA_CLRFR	659
PKA_CR	657
PKA_SR	658
PWR_C2CR1	187
PWR_C2CR3	189
PWR_CR1	172
PWR_CR2	173
PWR_CR3	174
PWR_CR4	176
PWR_CR5	180
PWR_EXTSCR	190
PWR_PDCRA	182
PWR_PDCRB	183
PWR_PDCRC	184
PWR_PDCRD	185
PWR_PDCRE	186
PWR_PDCRH	187
PWR_PUCRA	181
PWR_PUCRB	182
PWR_PUCRC	183
PWR_PUCRD	184
PWR_PUCRE	185
PWR_PUCRH	186
PWR_SCR	179
PWR_SR1	177
PWR_SR2	178

**Q**

QUADSPI_ABR	411
QUADSPI_AR	411
QUADSPI_CCR	409
QUADSPI_CR	404
QUADSPI_DCR	406
QUADSPI_DLR	409
QUADSPI_DR	412
QUADSPI_FCR	408
QUADSPI_LPTR	414
QUADSPI_PIR	413
QUADSPI_PSMAR	413
QUADSPI_PSMKR	412
QUADSPI_SR	407

**R**

RCC_AHB1ENR	245, 268
RCC_AHB1RSTR	237-238
RCC_AHB1SMENR	251, 275
RCC_AHB2ENR	246, 269
RCC_AHB2RSTR	239
RCC_AHB2SMENR	252, 276

RCC_AHB3ENR	247, 270	SAI_ACR2	1252
RCC_AHB3RSTR	240	SAI_ADR	1270
RCC_AHB3SMENR	254, 278	SAI_AFRCR	1256
RCC_APB1ENR1	248, 271	SAI_AIM	1261
RCC_APB1ENR2	250, 273-274, 282	SAI_ASLOTR	1259
RCC_APB1RSTR1	241	SAI_ASRR	1264
RCC_APB1RSTR2	243	SAI_BCLRFR	1269
RCC_APB1SMENR1	255, 279	SAI_BCR1	1250
RCC_APB1SMENR2	257, 280	SAI_BCR2	1254
RCC_APB2ENR	250, 273	SAI_BDR	1271
RCC_APB2RSTR	243-244	SAI_BFRCR	1258
RCC_APB2SMENR	257, 281	SAI_BIM	1263
RCC_BDCR	260	SAI_BSLOTR	1260
RCC_CCIPR	259	SAI_BSR	1266
RCC_CFGR	225	SAI_PDMCR	1271
RCC_CICR	236	SAI_PDMDLY	1272
RCC_CIER	233	SPIx_CR1	1202
RCC_CIFR	234	SPIx_CR2	1204
RCC_CR	221	SPIx_CRCPR	1208
RCC_CRRCR	264-266	SPIx_DR	1207
RCC_CSR	262	SPIx_RXCRCR	1208
RCC_ICSCR	224	SPIx_SR	1206
RCC_PLLCFG	228	SPIx_TXCRCR	1208
RCC_PLLSAI1CFG	231	SYSCFG_C2IMR1	320
RNG_CR	583	SYSCFG_C2IMR2	321
RNG_DR	585	SYSCFG_CFGR1	310
RNG_SR	584	SYSCFG_CFGR2	317
RTC_ALRMAR	937	SYSCFG_EXTICR1	311
RTC_ALRMASSR	948	SYSCFG_EXTICR2	312
RTC_ALRMBR	938	SYSCFG_EXTICR3	313
RTC_ALRMBSSR	949	SYSCFG_EXTICR4	315
RTC_BKPxR	950	SYSCFG_IMR1	319
RTC_CALR	944	SYSCFG_IMR2	319
RTC_CR	929	SYSCFG_MEMRMP	309
RTC_DR	928	SYSCFG_SCSR	316
RTC_ISR	932	SYSCFG_SIPCR	321
RTC_OR	950	SYSCFG_SKR	318
RTC_PRER	935	SYSCFG_SWPR1	318
RTC_SHIFTR	940	SYSCFG_SWPR2	318
RTC_SSR	939		
RTC_TAMPCCR	945		
RTC_TR	927		
RTC_TSDR	942	TIM1_AF1	755
RTC_TSSSR	943	TIM1_AF2	756
RTC_TSTR	941	TIM1_ARR	743
RTC_WPR	939	TIM1_BDTR	746
RTC_WUTR	936	TIM1_CCER	740
		TIM1_CCMR1	733-734
<b>S</b>		TIM1_CCMR2	737-738
SAI_ACLRFR	1268	TIM1_CCMR3	752
SAI_ACR1	1247	TIM1_CCR1	744
		TIM1_CCR2	745

TIM1_CCR3	745	TIMx_DCR	877
TIM1_CCR4	746	TIMx_DIER	864
TIM1_CCR5	753	TIMx_DMAR	878
TIM1_CCR6	754	TIMx_EGR	866
TIM1_CNT	743	TIMx_PSC	873
TIM1_CR1	722	TIMx_RCR	874
TIM1_CR2	723	TIMx_SR	865
TIM1_DCR	750	TPIU_ACPR	1496
TIM1_DIER	728	TPIU_CIDR0	1503
TIM1_DMAR	751	TPIU_CIDR1	1503
TIM1_EGR	732	TPIU_CIDR2	1503
TIM1_OR1	752	TPIU_CIDR3	1504
TIM1_PSC	743	TPIU CLAIMCLR	1499
TIM1_RCR	744	TPIU CLAIMSETR	1498
TIM1_SMCR	726	TPIU_CSPSR	1496
TIM1_SR	730	TPIU_DEVIDR	1499
TIM1_TISEL	758	TPIU_DEVTYPER	1500
TIM16_AF1	879	TPIU_FFCR	1497
TIM16_OR1	879	TPIU_FFSR	1497
TIM16_TISEL	880	TPIU_FSCR	1498
TIM17_AF1	881	TPIU_PIDR0	1501
TIM17_OR1	880	TPIU_PIDR1	1501
TIM17_TISEL	882	TPIU_PIDR2	1502
TIM2_AF1	828	TPIU_PIDR3	1502
TIM2_ARR	825	TPIU_PIDR4	1500
TIM2_CCER	822	TPIU_SPPR	1496
TIM2_CCMR1	816, 818	TPIU_SSPPSR	1495
TIM2_CCMR2	820-821	TSC_CR	564
TIM2_CCR1	825	TSC_ICR	567
TIM2_CCR2	825	TSC_IER	566
TIM2_CCR3	826	TSC_IOASCR	569
TIM2_CCR4	826	TSC_IOCCR	570
TIM2_CNT	823-824	TSC_IOGCSR	570
TIM2_CR1	806	TSC_IOGxCR	571
TIM2_CR2	807	TSC_IOHCR	568
TIM2_DCR	827	TSC_IOSCR	569
TIM2_DIER	812	TSC_ISR	568
TIM2_DMAR	828		
TIM2_EGR	815		
TIM2_OR1	828		
TIM2_PSC	824	USART_BRR	1103
TIM2_SMCR	809	USART_CR1	1087, 1091
TIM2_SR	813	USART_CR2	1094
TIM2_TISEL	829	USART_CR3	1098
TIMx_ARR	873	USART_GTPR	1103
TIMx_BDTR	875	USART_ICR	1117
TIMx_CCER	870	USART_ISR	1106, 1112
TIMx_CCMR1	867-868	USART_PRESC	1120
TIMx_CCR1	874	USART_RDR	1119
TIMx_CNT	872	USART_RQR	1105
TIMx_CR1	862	USART_RTOR	1104
TIMx_CR2	863	USART_TDR	1119

**U**

USART_BRR	1103
USART_CR1	1087, 1091
USART_CR2	1094
USART_CR3	1098
USART_GTPR	1103
USART_ICR	1117
USART_ISR	1106, 1112
USART_PRESC	1120
USART_RDR	1119
USART_RQR	1105
USART_RTOR	1104
USART_TDR	1119

USB_ADDRN_RX .....	1334
USB_ADDRN_TX .....	1333
USB_BCDR .....	1327
USB_BTABLE .....	1326
USB_CNTR .....	1320
USB_COUNTn_RX .....	1334
USB_COUNTn_TX .....	1333
USB_DADDR .....	1325
USB_EPnR .....	1328
USB_FNR .....	1325
USB_ISTR .....	1322
USB_LPMCSR .....	1326

**V**

VREFBUF_CCR .....	509
VREFBUF_CSR .....	508

**W**

WWDG_CFR .....	966
WWDG_CR .....	965
WWDG_SR .....	967

**IMPORTANT NOTICE – READ CAREFULLY**

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to [www.st.com/trademarks](http://www.st.com/trademarks). All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2022 STMicroelectronics – All rights reserved